T·L·G
Texts in Logic and Games
Volume 1

Interactive Logic

Selected Papers from the 7th Augustus de Morgan Workshop, London

EDITED BY
JOHAN VAN BENTHEM, DOV GABBAY, BENEDIKT LÖWE

AMSTERDAM UNIVERSITY PRESS

INTERACTIVE LOGIC

Interactive Logic

Selected Papers from the 7th Augustus de Morgan Workshop, London

EDITED BY
JOHAN VAN BENTHEM
DOV GABBAY
BENEDIKT LÖWE

Texts in Logic and Games
Volume 1

AMSTERDAM UNIVERSITY PRESS

Cover design: Maedium, Utrecht

ISBN 978 90 5356 356 4
NUR 918

Table of Contents

Preface

Logic has many faces. Some of them look toward abstract propositions and meanings, underpinned by the notion of *truth* as quiet reflection of reality within language. In this view, logic involves no agency at all, as propositions are true or false for all eternity.

But perhaps the central theme of logic since Aristotle is that of *inference*, which intuitively involves one (idealized) agent doing derivations and inspecting arguments in his or her mind.

In the last decades, many-agent activities have come to the fore. A major research focus nowadays is the field of *interactive logic*, studying communication, argumentation, and intelligent interaction in general. These new developments include dynamic logic, old and new epistemic logics, logics of games, social choice, and other areas. Parikh has coined the phrase *social software* for the investigation of the formal structure of social procedures with the means of logic and related areas of computer science.

But at the end of the day, this latest trend is also a return to the sources. Logic in both its Western and Eastern origins started from the study of argumentation, communication, and legal procedure, as is amply demonstrated by early texts on the law of non-contradiction and other inferential phenomena.

The 7th Augustus de Morgan Workshop was held at King's College London from November 4th to 7th, 2005 under the title *INTERACTIVE LOGIC: Games and Social Software.* It brought together a lively community of logicians working on various key aspects of the interactive turn. The present volume is a collection of fully refereed papers based on the presentations at the workshop.

The Augustus De Morgan Workshops

The Augustus De Morgan Workshops address interdisciplinary areas concerning logics devised and used to model human reasoning, actions and behaviour. Such models arise in various practical and theoretical areas striving to provide devices that help/replace the human in his daily activity and require, by their very nature, an interdisciplinary cooperation involving mathematical logic, automated deduction, knowledge representation and reasoning, artificial intelligence, multi-agent systems, natural language processing, philosophy, linguistics, law, etc.

In 1999, a plan was made to hold an Augustus De Morgan Workshop every year, devoted to an important interdisciplinary applied logic area and to invite some of the best and most active researchers in this area to focus on some clear problems. These workshops honour Augustus De Morgan (1806–1871) who was a Professor of Mathematics at London University.

The first six Augustus De Morgan Workshops took place in the years 1999 to 2004 and covered topics from history of logic via belief revision to Logic & Law.

ADMW 2005 and this volume

The seventh Augustus de Morgan Workshop was a close collaboration of the logicians at King's College London and at the Institute for Logic, Language and Computation (ILLC) in Amsterdam.

The organizers planned the event in the tradition of the conference series TARK and LOFT; and for many participants, ADMW 2005 (London, November 2005), LOFT 2006 (Liverpool, July 2006), and the workshop *New Perspectives on Games and Interaction* at the Royal Academy in Amsterdam (February 2007) constituted a natural sequence of events in the field of interactive logic. It is fitting that the proceedings of the other two events will also appear in the same new book series, *Texts in Logic and Games.*

This book series of which this is the first volume was conceived of in London during ADMW 2005. The Managing Editors to be met to discuss the set-up of the series and quickly found eminent members for the future Advisory Board. We are proud to have the present ADMW 2005 proceedings as the inaugural volume of the new series, which will hopefully become a focus in the area.

This volume contains eleven original research contributions that were reviewed to the high standards of an international research journal. Twenty-four referees helped us to review the twenty-one submissions and select twelve of them for the volume. In many cases, the referee reports were very detailed and led to considerable improvements of the papers. We would like to thank all anonymous referees for their support—the quality of this volume is testimony to their efforts.

Acknowledgements

The local organization of the workshop was single-handedly managed by Jane Spurr for which we would like to express our most sincere gratitude. The production phase of this volume was coordinated by the technical assistant of the book series *Texts in Logic and Games*, Joel Uckelman, who wrote the LaTeX stylefile for TLG, served as the technical contact point for the au-

thors and put in countless hours during the final typesetting phase. Finally, we would like to thank the *Societas Philosophica Fennica* and Tuomo Aho and Ahti-Veikko Pietarinen for the permission to reprint Samson Abramsky's paper in this volume.

Amsterdam & London J.F.A.K.v.B. D.G. B.L.

Participants

Samson Abramsky (*Oxford, England*), Krzysztof Apt (*Amsterdam, The Netherlands*), Johan van Benthem (*Amsterdam, The Netherlands*), Dietmar Berwanger (*Aachen, Germany*), Stefano Borgo (*Trento, Italy*), Julian Bradfield (*Edinburgh, Scotland*), Adam Brandenburger (*New York NY, United States of America*), J. Robin Cameron (*Aberdeen, Scotland*), Daniel Eckert (*Graz, Austria*), Jan van Eijck (*Amsterdam, The Netherlands*), Corinna Elsenbroich (*London, England*), Dov Gabbay (*London, England*) Artur d'Avila Garcez (*London, England*), Andrew Gargett (*London, England*), Konrad Grabiszewski (*New York NY, United States of America*), Erich Grädel (*Aachen, Germany*), Stephan Hartmann (*London, England*), Wilfrid Hodges (*London, England*), Wiebe van der Hoek (*Liverpool, England*), Hykel Hosni (*Manchester, England*), Ruth Kempson (*London, England*), Vladimir Komendantsky (*Cork, Ireland*), Roman Kontchakov (*London, England*), Barteld Kooi (*Groningen, The Netherlands*), Luis Lamb (*London, England*), Christian List (*London, England*), Benedikt Löwe (*Amsterdam, The Netherlands*), David Makinson (*London, England*), Wilfried Meyer-Viol (*London, England*), Maxime Morge (*Lille, France*), Eric Pacuit (*Amsterdam, The Netherlands*), Xavier Parent (*London, England*), Rohit Parikh (*New York NY, United States of America*), Andrés Perea (*Maastricht, The Netherlands*), Gabriella Pigozzi (*London, England*), Robert van Rooij (*Amsterdam, The Netherlands*), Olivier Roy (*Amsterdam, The Netherlands*), Mehrnoosh Sadrzadeh (*Montréal QC, Canada*), Peter Schroeder-Heister (*Tübingen, Germany*), Brian Semmes (*Amsterdam, The Netherlands*), Merlijn Sevenster (*Amsterdam, The Netherlands*), Valentin Shehtman (*London, England*), Matthew Stone (*Edinburgh, Scotland*), Andrjez Szalas (*London, England*), Aarne Talman (*London, England*), Tero Tulenheimo (*Helsinki, Finland*), Sara Uckelman (*Amsterdam, The Netherlands*), Joel Uckelman (*Amsterdam, The Netherlands*), Jouko Väänänen (*Helsinki, Finland*), Davi Romero de Vasconcelos (*Rio de Janeiro, Brazil*), Philip Welch (*Bristol, England*), John Woods (*Vancouver BC, Canada*), Adam Wyner (*London, England*), Tomoyuki Yamada (*Sapporo, Japan*).

A Compositional Game Semantics for Multi-Agent Logics of Partial Information[*]

Samson Abramsky

Computing Laboratory
Oxford University
Wolfson Building, Parks Road
Oxford OX1 3QD, United Kingdom
samson@comlab.ox.ac.uk

Abstract

We consider the following questions: What kind of logic has a natural semantics in multi-player (rather than 2-player) games? How can we express branching quantifiers, and other partial-information constructs, with a properly compositional syntax and semantics? We develop a logic in answer to these questions, with a formal semantics based on multiple concurrent strategies, formalized as closure operators on Kahn-Plotkin concrete domains. Partial information constraints are represented as co-closure operators. We address the syntactic issues by treating syntactic constituents, including quantifiers, as arrows in a category, with arities and co-arities. This enables a fully compositional account of a wide range of features in a multi-agent, concurrent setting, including IF-style quantifiers.

1 Introduction

We begin with the following quote from the manifesto of the 7th Augustus de Morgan workshop:

> Traditionally, logic has dealt with the zero-agent notion of truth and the one-agent notion of reasoning. In the last decades, research focus in logic shifted from these topics to the vast field of "interactive logic", encompassing logics of communication and interaction. The main applications of this move to n-agent notions are logical approaches to games and social software.

However, while there are certainly applications of multi-modal logics to reasoning *about* n-person games (see e.g. [$\text{Pa}_7\text{Pa}_5$03, $\text{Pa}_7$02]), the more

[*] This paper is an updated and reprinted version of *"Socially Responsive, Environmentally Friendly Logic"*, published in [$\text{Ah}_1\text{Pi}_1$06, pp. 17–45]. The author and the volume editors would like to thank the copyright holder, the Philosophical Society of Finland, for their kind permission to reprint the paper in this volume.

intimate connections between Games and Logic which manifest themselves in various forms of Game Semantics have all, to the best of our knowledge, been based on 2-person games. We are therefore led to consider the following question:

> What kind of logic has a natural semantics in multi-player (rather than 2-player) games?

Another topic which has been studied extensively in recent years has been the logical aspects of games of imperfect information, starting with Henkin-style branching quantifiers [$\mathrm{He}_1$61], and Hintikka's game-theoretical interpretation of these, and continuing with the IF-logic of Hintikka and Sandu [$\mathrm{Hi}_1\mathrm{Sa}_4$89, $\mathrm{Hi}_1\mathrm{Sa}_4$95, $\mathrm{Hi}_1\mathrm{Sa}_4$96]. The issue of whether and how a compositional semantics for IF-logic can be given has been studied by several authors, particularly Wilfrid Hodges [$\mathrm{Ho}_1$97a]. However, there is an even more basic question which does not seem to have received much, if any, attention: namely, how to give a properly compositional *syntax* for such constructs. For example, how can we build up a formula with branching quantifiers piece by piece? It might seem that IF-logic sweeps this question aside, since it does on the face of it have a compositional syntax. However, more careful consideration shows that the scope issues raised by the IF quantifiers and connectives do not fit into the usual pattern of variable-binding operators.

Our aim in the present paper is to develop a logical syntax, and an accompanying formal semantics, which addresses both these questions. The semantics is naturally phrased in terms of strategies for n-person games; and it will fit well with our compositional analysis of partial information constructs. Both our syntactical and semantical explorations will bring to light some rather unexpected connections to developments in Theoretical Computer Science.

Many of the ideas in this paper were first presented in a lecture given at the 11th Amsterdam Colloquium in December 1997. Some of the underlying technical notions are developed in rather different contexts in a number of other papers [$\mathrm{AbJa}_1$94, $\mathrm{AbMe}_1$99, Ab00a, Ab03]. One motivation for writing this paper is to attempt to communicate some of the techniques and concepts which have been developed within the Theoretical Computer Science semantics community to a broader audience. We have therefore tried to present the ideas in a self-contained fashion, and in a fairly expansive expository style. The present paper is a revised and expanded version of a paper which first appeared as [Ab06].

2 From 2-person to n-person games

2.1 The naturalness of 2-person games

The basic metaphor of Game Semantics of Logic is that the players stand for **Proponent** and **Opponent**, or **Verifier** and **Falsifier**, or (with Computer Science motivation) for **System** and **Environment**. This 2-agent view sets up a natural *duality*, which arises by interchanging the rôles of the two players. This duality is crucial in defining the key notion of *composition of strategies* in the Game Semantics developed in Computer Science. It stands as the Game-Semantical correlate of the logical notion of *polarity*, and the categorical notions of *domain* and *codomain*, and *co-* and *contra-variance.*

So this link between Logic and 2-person games runs deep, and should make us wary of facile generalization. It can be seen as having the same weight as the binary nature of composition in Categories, or of the Cut Rule in Logic. Are there good multi-ary generalizations of these notions?

Nevertheless ... we shall put forward a simple system which seems to us to offer a natural generalization. We shall validate this notion to the extent of providing a precise and (in our opinion) elegant semantics in n-person games. This at least has the merit of broaching the topic, and putting a clear, if far from comprehensive, proposal on the table. There are undoubtedly many further subtleties to explore, but it is a start.

2.2 Background: 2-person games

As a starting point, we shall briefly review a Hintikka-style 'Game-Theoretical Semantics' of ordinary first-order logic, in negation normal form. Thus formulas are built from literals by conjunction, disjunction, and universal and existential quantification. Given a model $\mathcal{M}$, a game is assigned to each sentence as follows. (We shall be informal here, just as Hintikka invariably is.)

- **Literals** $A(a_1, \ldots, a_n)$, $\neg A(a_1, \ldots, a_n)$. The game is trivial in this case. There is a winning move for Verifier if the literal is true in the model, and a winning move for Falsifier otherwise.
- **Conjunction** $\varphi_1 \wedge \varphi_2$. The game $G(\varphi_1 \wedge \varphi_2)$ has a first move by Falsifier which chooses one of the sub-formulas φ_i, $i = 1, 2$. It then proceeds with $G(\varphi_i)$.
- **Disjunction** $G(\varphi_1 \vee \varphi_2)$ has a first move by Verifier which chooses one of the sub-formulas φ_i, $i = 1, 2$. It then proceeds with $G(\varphi_i)$.
- **Universal Quantification** $G(\forall x. \varphi)$ has a first move by Falsifier, which chooses an element a of $\mathcal{M}$. The game proceeds as $G(\varphi[a/x])$.
- **Existential Quantification** Dually, Verifier chooses the element.

The point of this interpretation is that $\mathcal{M} \models \varphi$ in the usual Tarskian sense if and only if Verifier has a winning strategy for $G(\varphi)$.

Note that there is a very natural game-semantical interpretation of negation: $G(\neg\varphi)$ is the same game as $G(\varphi)$, but with the rôles of Verifier and Falsifier interchanged.

2.3 An aside

In fact, the above Game semantics should really be seen as applying to the *additive fragment of Linear Logic*, rather than to Classical Logic. Note in particular that it fails to yield a proper analysis of *implication*, surely the key logical connective.

Indeed, if we render $\varphi \rightarrow \psi$ as $\neg\varphi \ \vee \ \psi$, then note that $G(\neg\varphi \ \vee \ \psi)$ does not allow for any flow of information between the antecedent and the consequent of the implication. At the very first step, one of $\neg\varphi$ or ψ is chosen, and the other is discarded.[1] In order to have the possibility of such information flow, it is necessary for $G(\neg\varphi)$ and $G(\psi)$ to run *concurrently.* This takes us into the realm of the *multiplicative connectives* in the sense of Linear Logic [Gi$_2$87]. The game-theoretical interpretation of negation also belongs to the multiplicative level.

2.4 From 2-person to n-person games

We shall now describe a simple syntax which will carry a natural interpretation in n-agent games. We say that the resulting logic is "socially responsive" in that it allows for the actions of multiple agents.

We fix, once and for all, a set of agents $\mathcal{A}$, ranged over by $\alpha, \beta, \ldots$

We introduce an $\mathcal{A}$-indexed family of binary connectives $\oplus_\alpha$, and an $\mathcal{A}$-indexed family of quantifiers Q_α. Thus we have a syntax:

$$\varphi \quad ::= \quad L \ \mid \ \varphi \oplus_\alpha \psi \ \mid \ Q_\alpha x.\, \varphi.$$

(Here L ranges over literals.)

The intended interpretation of $\varphi \oplus_\alpha \psi$ is a game in which agent α initially chooses either φ or ψ, and then play proceeds in the game corresponding to the chosen sub-formula. Similarly, for $Q_\alpha x.\, \varphi$, α initially chooses an instance a for x, and play proceeds as for $\varphi[a/x]$.

2.4.1 2-person games as a special case

If we take $\mathcal{A} = \{V, F\}$, then we can make the following identifications:

$$\oplus_V = \vee, \qquad \oplus_F = \wedge, \qquad Q_V = \exists, \qquad Q_F = \forall.$$

[1] This is of course quite analogous to the "paradoxes of material implication". Our point is that the game structure, if taken seriously as a way of articulating interactive behaviour, rather than merely being used as a carrier for standard model-theoretic notions, opens up new and more interesting possibilities.

2.4.2 Whither negation?

In 2-person Game Semantics, negation is interpreted as rôle interchange. This generalizes in the multi-agent setting to *rôle permutation*. Each permutation $\pi \in S(\mathcal{A})$ ($S(X)$ being the symmetric group on the set X) induces a logical operation $\hat{\pi}(\varphi)$ of permutation of the rôles of the agents in the game corresponding to A. In the 2-agent cases, there are two permutations in $S(\{V, F\})$, the identity (a 'no-op'), and the transposition $V \leftrightarrow F$, which corresponds exactly to the usual game-theoretical negation.

2.4.3 Other connectives

In the light of our remarks about the essentially *additive* character (in the sense of Linear Logic) of the connectives $\oplus_\alpha$ and their game-semantical interpretation, it is also natural to consider some multiplicative-style connectives. We shall introduce two very basic connectives of this kind, which will prove particularly useful for the compositional analysis of branching quantifiers:

1. **Parallel Composition**, $\varphi \| \psi$. The intended semantics is that $G(\varphi \| \psi)$ is the game in which play in $G(\varphi)$ and $G(\psi)$ proceeds in parallel.
2. **Sequential Composition**, $\varphi \cdot \psi$. Here we firstly play in $G(\varphi)$ to a conclusion, and then play in $G(\psi)$.

It will also be useful to introduce a constant $\mathbf{1}$ for a "neutral" or "vacuously true" proposition. The intended semantics is an empty game, in which nothing happens. Thus we should expect $\mathbf{1}$ to be a unit for both sequential and parallel composition.

Thus the syntax for our multi-agent logic $\mathcal{L}_\mathcal{A}$ stands as follows:

$$\varphi \quad ::= \quad \mathbf{1} \;\mid\; A \;\mid\; \varphi \oplus_\alpha \psi \;\mid\; Q_\alpha x.\, \varphi \;\mid\; \varphi \cdot \psi \;\mid\; \varphi \| \psi \;\mid\; \hat{\pi}(\varphi).$$

Here A ranges over atomic formulas, and $\pi \in S(\mathcal{A})$.

2.4.4 Quantifiers as particles

The syntax of $\mathcal{L}_\mathcal{A}$ is more powerful than may first appear. Consider an idea which may seem strange at first, although it has also arisen in Dynamic Game Logic [vB03] and implicitly in the semantics of non-deterministic programming languages [ApPl$_1$81]. Instead of treating quantifiers as prefixing operators $Q_\alpha x.\varphi$ in the usual way, we can consider them as stand-alone particles $Q_\alpha x \equiv Q_\alpha x.\mathbf{1}$. We should expect to have

$$(Q_\alpha x) \cdot \varphi \;\equiv\; (Q_\alpha x.\mathbf{1}) \cdot \varphi \;\equiv\; Q_\alpha x.(\mathbf{1} \cdot \varphi) \;\equiv\; Q_\alpha x.\, \varphi. \qquad (1.1)$$

Thus this particle view of quantifiers does not lose any generality with respect to the usual syntax for quantification. But we can also express much more:

$$[(\forall x \exists y) \;\|\; (\forall u \exists v)] \cdot A(x, y, u, v).$$

This is the **Henkin quantifier**, expressed compositionally in the syntax of $\mathcal{L}_\mathcal{A}$.[2] More generally:

Proposition 2.1. Every partially-ordered quantifier prefix in which the partial order is a series-parallel poset can be expressed in the syntax of $\mathcal{L}_\mathcal{A}$.

We could therefore recast the grammar of $\mathcal{L}_\mathcal{A}$ as follows:

$$\varphi \quad ::= \quad A \ \mid \ \varphi \oplus_\alpha \psi \ \mid \ Q_\alpha x \ \mid \ \varphi \cdot \psi \ \mid \ \varphi \| \psi \ \mid \ \hat{\pi}(\varphi).$$

However, we shall stick to the previous syntax, as this is more familiar, and will provide us with an independent check that the expected equivalences (1.1) hold.

3 Semantics of $\mathcal{L}_\mathcal{A}$

We shall now develop a semantics for this logic. This semantics will be built in two levels:

1. **Static Semantics of Formulas** To each formula $\varphi \in \mathcal{L}_\mathcal{A}$, we shall assign a form of game rich enough to allow for concurrent actions, and for rather general forms of temporal or causal dependency between moves. This assignment will be fully compositional.

2. **Dynamic Semantics** We then formulate a notion of *strategy* for these games. For each agent $\alpha \in \mathcal{A}$ there will be a notion of α-strategy. We shall show to build strategies for the games arising from formulas, compositionally from the strategies for the sub-formulas. We shall also define the key notion of how to evaluate *strategy profiles*, *i.e.* a choice of strategy for each agent, interacting with each other to reach a collective outcome. We shall find an elegant mathematical expression for this, *prima facie* very complicated, operational notion.

   We shall also discuss the notion of *valuation of an outcome*, on the basis of which logical notions of *validity*, or game-theoretical notions of *equilibria* can be defined. However, we shall find that a fully compositional account of valuations will require the more refined analysis of syntax to be given in the next section.

3.1 Static semantics: concrete data structures as concurrent games

The structures we shall find convenient, and indeed very natural, to use as our formal representations of games were introduced by Gilles Kahn and Gordon Plotkin in 1975 (although their paper only appeared in a journal

[2] A similar analysis of branching quantifier syntax appears in the appendix to [vB03]; my thanks to Johan van Benthem for pointing out this reference to me.

in 1993; *cf.* [Ka$_0$Pl$_1$78, Ka$_0$Pl$_1$93]). They arose for Kahn and Plotkin in providing a representation theory for their notion of *concrete domains.* The term used by Kahn and Plotkin for these structures was *information matrices*; subsequently, they have usually been called *concrete data structures*, and we shall follow this latter terminology (although in some ways, the original name is more evocative in our context of use).

What, then, is a concrete data structure (CDS)? It is a structure

$$M = (C, V, D, \vdash)$$

where:

- C is a set of *cells*, or 'loci of decisions'—places where the agent can make their moves. These cells have a spatio-temporal significance: they both allow the distributed nature of multi-agent interactions to be articulated, and also capture a causal or temporal flow between events, as we shall see.

- V is a set of 'values', which label the choices which can be made by the agents from their menus of possible moves: for example, choosing the first or second branch of a compound formula $\varphi \oplus_\alpha \psi$, or choosing an instance for a quantifier.

- $D \subseteq C \times V$ is a set of *decisions*, representing the possible choices which can be made for how to 'fill' a cell. Note that specifying D allows some primitive typing for cells; only certain choices are appropriate for each given cell. (The more usual terminology for decisions is 'events'; here we have followed Kahn and Plotkin's original terminology, which is very apt for our purposes.)

- The relation $\vdash \subseteq \mathcal{P}_\mathsf{f}(D) \times C$ is an *enabling relation* which determines the possible temporal flows of events in a CDS. ($\mathcal{P}_\mathsf{f}(X)$ is the set of *finite* subsets of X.)

A *state* or *configuration* over a CDS M is a set $s \subseteq D$ such that:

- s is a partial function, *i.e.* each cell is filled at most once.

- If $(c, v) \in s$, then there is a sequence of decisions

  $$(c_1, v_1), \ldots, (c_k, v_k) = (c, v)$$

  in s such that, for all j, $1 \leq j \leq k$, for some $\Gamma_j \subseteq \{(c_i, v_i) \mid 1 \leq i < j\}$:

  $$\Gamma_j \vdash c_j.$$

  This is a "causal well-foundedness" condition. (Kahn and Plotkin phrase it as: "*c has a proof in x*".) Note that, in order for there to be any non-empty states, there must be *initial cells* c_0 such that $\varnothing \vdash c_0$.

We write $\mathcal{D}(M)$ for the set of states, partially ordered by set inclusion. This is a concrete domain in the sense of Kahn and Plotkin. In particular, it is closed under directed unions and unions of bounded families, and the finite states form a basis of compact elements, so it is algebraic.

To obtain a structure to represent a multi-agent game, we shall consider a CDS M augmented with a *labelling map*

$$\lambda_M : C_M \longrightarrow \mathcal{A}$$

which indicates which agent is responsible for filling each cell. We call (M, λ_M) an $\mathcal{A}$*-game.*

We are now ready to specify the compositional assignment of an $\mathcal{A}$-game to each formula of $\mathcal{L}_{\mathcal{A}}$. We assume a set $\mathcal{I}$ which will be used as the domain of quantification. We shall use $\uplus$ for the disjoint union of sets.

- **Constant 1**. This is assigned the empty CDS $(\varnothing, \varnothing, \varnothing, \varnothing)$.
- **Atomic formulas** A. These are assigned the empty CDS $(\varnothing, \varnothing, \varnothing, \varnothing)$.
- **Choice connectives** $\varphi \oplus_\alpha \psi$. Let

  $$M = [\![\varphi]\!], \qquad N = [\![\psi]\!]$$

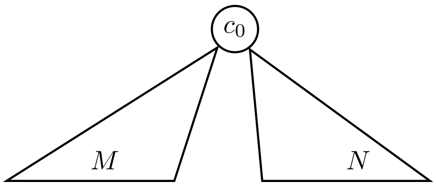
  be the CDS assigned to φ and ψ, with labelling functions λ_M and λ_N. Then the CDS $M \oplus_\alpha N$ is defined as follows:

  $$(C_M \uplus C_N \uplus \{c_0\}, V_M \cup V_N \cup \{1,2\}, D_M \uplus D_N \uplus \{(c_0,1),(c_0,2)\}, \vdash_{M \oplus_\alpha N})$$

  where

  $$\begin{array}{l} \vdash_{M\oplus_\alpha N} c_0 \\ (c_0,1), \Gamma \vdash_{M\oplus_\alpha N} c \iff \Gamma \vdash_M c \\ (c_0,2), \Gamma \vdash_{M\oplus_\alpha N} c \iff \Gamma \vdash_N c. \end{array}$$

  This is the standard *separated sum* construction on CDS as in [Ka$_0$Pl$_1$93]. Pictorially:



  Initially, only the new cell c_0 is enabled. It can be filled with either of the values 1 or 2. If it is filled by 1, we can proceed as in $M = [\![\varphi]\!]$, while if it is filled with 2, we proceed as in $N = [\![\psi]\!]$. This makes the usual informal specification precise, in a rather general setting.

To complete the specification of $M \oplus_\alpha N$ as an $\mathcal{A}$-game, we specify the labelling function $\lambda_{M \oplus_\alpha N}$:

$$\begin{array}{rcll} c_0 & \mapsto & \alpha & \\ c & \mapsto & \lambda_M(c) & (c \in C_M) \\ c & \mapsto & \lambda_N(c) & (c \in C_N). \end{array}$$

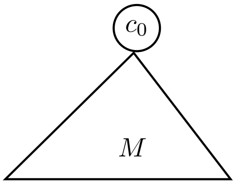
As expected, the initial cell c_0 must be filled by the agent α; other cells are filled as in the corresponding sub-games.

- **Quantifiers** $Q_\alpha x.\,\varphi$. Let $M = [\![\varphi]\!]$.

$$Q_\alpha(M) \;=\; (C_M \uplus \{c_0\}, V_M \cup \mathcal{I}, D_M \uplus (\{c_0\} \times \mathcal{I}), \vdash_{Q_\alpha(M)}).$$

$$\begin{array}{l} \vdash_{Q_\alpha(M)} c_0 \\ (c_0, a), \Gamma \vdash_{Q_\alpha(M)} c \iff \Gamma \vdash_M c \quad (a \in \mathcal{I}). \end{array}$$

This is a variant of the standard *lifting* construction on CDS.



Initially, only the new cell c_0 is enabled. It can be filled with any choice of individual a from the domain of quantification $\mathcal{I}$. Subsequently, we play as in M. The labelling function, $\lambda_{Q_\alpha(M)}$:

$$c_0 \mapsto \alpha, \qquad c \mapsto \lambda_M(c) \quad (c \in C_M).$$

Although the interpretation of the quantifier particle $Q_\alpha x \equiv Q_\alpha x.\,\mathbf{1}$ can be derived from the definitions already given, we set it out explicitly to show how simple and natural it is:

$$[\![Q_\alpha x]\!] = (\{c_0\}, \mathcal{I}, \{c_0\} \times \mathcal{I}, \{\vdash c_0\}),$$

with labelling function $c_0 \mapsto \alpha$. Thus the game consists of a single cell, labelled by α, initially enabled, in which any element from $\mathcal{I}$ can be chosen — a true "particle of action" in a multi-agent setting.

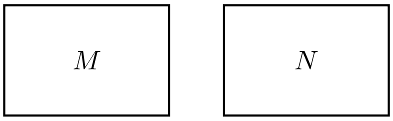
- **Parallel Composition** $\varphi \| \psi$. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$: we define

$$M \| N \quad = \quad (C_M \uplus C_N, V_M \uplus V_N, D_M \uplus D_N, \vdash_M \uplus \vdash_N).$$

The labelling function is defined by:

$$\lambda_{M\|N}(c) = \begin{cases} \lambda_M(c) & (c \in C_M) \\ \lambda_N(c) & (c \in C_N). \end{cases}$$

Pictorially:



Decisions in M and N can be made concurrently, with no causal or temporal constraints between them. This is the standard *product* construction on CDS.

- **Sequential Composition** $\varphi \cdot \psi$. We say that a state $s \in \mathcal{D}(M)$ is *maximal* if

$$\forall t \in \mathcal{D}(M)[s \subseteq t \;\Rightarrow\; s = t].$$

We write $\mathsf{Max}(M)$ for the set of maximal elements of $\mathcal{D}(M)$.

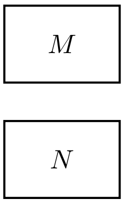
Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$: we define

$$M \cdot N \quad = \quad (C_M \uplus C_N, V_M \uplus V_N, D_M \uplus D_N, \vdash_{M \cdot N}),$$

where

$$\Gamma \vdash_{M \cdot N} c \quad \Longleftrightarrow \quad \Gamma_M \vdash c \;\;\vee\;\; (\Gamma = s \cup \Delta \wedge s \in \mathsf{Max}(M) \wedge \Delta \vdash_N c).$$

Pictorially:



The idea is that firstly we reach a maximal state in M—a "complete play"—and then we can continue in N. Note that this construction makes sense for arbitrary CDS M, N. Even if M has infinite maximal states, the finitary nature of the enabling relation means that no events from N can occur in $M \cdot N$ following an infinite play in M.

The labelling function is defined by:

$$\lambda_{M \cdot N}(c) = \begin{cases} \lambda_M(c) & (c \in C_M) \\ \lambda_N(c) & (c \in C_N). \end{cases}$$

Note that the difference between $M \| N$ and $M \cdot N$ is purely one of *temporality* or *causality*: when events can occur (or, in the alternative terminology, decisions can be made) relative to each other.

- **Role Switching** $\hat{\pi}(\varphi)$, $\pi \in S(\mathcal{A})$. The CDS $[\![\hat{\pi}(\varphi)]\!]$ is *the same* as $M = [\![\varphi]\!]$. However:
$$\lambda_{\hat{\pi}(M)} = \pi \circ \lambda_M.$$

3.2 Dynamic semantics: concurrent strategies

We now turn to the task of defining a suitable notion of *strategy* for our games. We shall view a strategy for an agent α on the game M as a function $\sigma : \mathcal{D}(M) \to \mathcal{D}(M)$. The idea is that $\sigma(s)$ shows the moves which agent α would make, in the situation represented by the state s, when following the strategy represented by σ. Some formal features of σ follow immediately from this:

- (S1) Since past moves cannot be undone, we must have $s \subseteq \sigma(s)$, *i.e.* σ is *increasing.*
- (S2) If $(c, v) \in \sigma(s) \setminus s$, it must be the case that $\lambda_M(c) = \alpha$, since α is only able to make decisions in its own cells.

We shall impose two further conditions. While not quite as compelling as the two above, they also have clear motivations.

- (S3) Idempotence: $\sigma(\sigma(s)) = \sigma(s)$. Since the only information in $\sigma(s)$ over and above what is in s is what σ put there, this is a reasonable normalizing assumption. It avoids situations where σ proceeds ‘slowly’, making decisions in several steps which it could have taken in one step (since the information from the Environment, *i.e.* the other agents, has not changed).
- (S4) Monotonicity: $s \subseteq t \;\Rightarrow\; \sigma(s) \subseteq \sigma(t)$. This condition reflects the idea that states only contain *positive* information. The fact that a cell c has *not* been filled yet is not a definite, irrevocable piece of information. Another strategy, working on behalf of another agent, may be running concurrently with us, and just about to fill c.

Finally, there is a more technical point, which is a necessary complement to the above conditions. We take σ to be a function on $\mathcal{D}(M)^\top$, which is obtained by adjoining a top element $\top$ to $\mathcal{D}(M)$. Note that, since σ is increasing, we must have $\sigma(\top) = \top$. The significance of adding a top element to the codomain is that it allows for *partial strategies*, which are undefined in some situations.

The following result is standard.

Proposition 3.1. $\mathcal{D}(M)^\top$ is an algebraic complete lattice.

Taking the conditions (S1), (S3), (S4) together says that σ is a *closure operator* on $\mathcal{D}(M)^\top$. A closure operator additionally satisfying (S2) is said to be an α-closure operator. We write $\mathsf{Cl}_\alpha(M)$ for the set of α-closure operators on $\mathcal{D}(M)^\top$.

The full specification of the game based on a CDS M will comprise a set of strategies $S_\alpha(M) \subseteq \mathsf{Cl}_\alpha(M)$ for each agent α. By limiting the set of strategies suitably, we can in effect impose constraints on the information available to agents.

3.2.1 Inductive construction of strategy sets

There are several approaches to defining the strategy sets S_α. We are interested in *compositional definitions* of $S_\alpha(\llbracket\varphi\rrbracket)$. There are two main approaches to such definitions, both of which have been extensively deployed in Game Semantics as developed in Computer Science.

1. We can define the strategy sets themselves directly, by induction on the construction on φ. This is the "global" approach. It is akin to realizability, and in general leads to strategy sets of high logical complexity. See [AbMe$_1$99, Ab00b] for examples of this approach.

2. We can use an indirect, more "local" approach, in which we add some structure to the underlying games, and use this to state conditions on strategies, usually phrased as conditions on individual plays or runs of strategies. $S_\alpha(\llbracket\varphi\rrbracket)$ is then defined to be the set of strategies in $\mathsf{Cl}_\alpha(\llbracket\varphi\rrbracket)$ satisfying these conditions. This has in fact been the main approach used in the Game Semantics of programming languages [AbJa$_1$Ma$_1$00, Hy$_0$On00]. However, this approach has not been developed for the kind of concurrent, multi-agent games being considered here.

It seems that both of these approaches may be of interest in the present context. We therefore show how both can be applied to the semantics of $\mathcal{L}_\mathcal{A}$. We begin with the local approach, which is perhaps closer to the intuitions.

3.2.2 Local conditions on strategies

The idea is to capture, as part of the structure of the underlying game, which information about the current state should be available to agent α when it makes a decision at cell c. This can be formalized by a function

$$\gamma_M : C_M \longrightarrow [\mathcal{D}(M) \longrightarrow \mathcal{D}(M)]$$

which for each cell c assigns a function $\gamma_M(c)$ on states. The idea is that, if $\lambda_M(c) = \alpha$, $\gamma_M(c)(s)$ restricts s to the part which should be visible to agent α, and on the basis of which he has to decide how to fill c. It follows that $\gamma_M(c)$ should be *decreasing*: $\gamma_M(c)(x) \subseteq x$. Note the duality of this condition to (S1). We add the assumptions of monotonicity and idempotence, with much the same motivation as for strategies. It follows that $\gamma_M(c)$ is a *co-closure operator*.

3.2.3 Notation

Remembering that a state $s \in \mathcal{D}(M)$ is a partial function, we write $s{\searrow}c$ to mean that s is defined at the cell c, or that "s fills c", as it is usually expressed. Also, we write C_M^α for the set of α-labelled cells in C_M.

Now, given such a function γ_M, we can define the strategy set $S_\alpha(M)$:

$$\begin{aligned} S_\alpha(M) = \{\sigma \in \mathsf{Cl}_\alpha(M) \mid & \\ & \forall s \in \mathcal{D}(M).\forall c \in C_M^\alpha.[\sigma(s){\searrow}c \;\Rightarrow\; \sigma(\gamma_M(c)(s)){\searrow}c]\}. \quad (1.2)\end{aligned}$$

Thus the information constraint imposed by γ_M is expressed by the condition that σ can only make a decision at cell c in state s if it would have made the same decision in the smaller (less information) state $\gamma_M(c)(s)$.[3] This is a direct analogue of the use of views in Hyland-Ong style games [Hy$_0$On00] to define 'innocent strategies'. However, apart from the more general format of our games, there is greater flexibility in the provision of the function γ_M as a separate component of the game structure, whereas specific view functions are built into the fabric of HO-games.

We now show how the functions $\gamma_{[\![\varphi]\!]}$ can be defined, compositionally in φ.

- **Atomic formulas, and constant 1**. This case is trivial, since the set of cells is empty.
- **Choice connectives** $\varphi \oplus_\alpha \psi$. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$. We define $\gamma_{M\oplus_\alpha N}$ by:

$$\gamma_{M\oplus_\alpha N}(c)(s) = \begin{cases} \varnothing, & c = c_0 \\ \{(c_0, 1)\} \cup \gamma_M(c)(s \setminus (c,1)), & (c \in C_M) \\ \{(c_0, 2)\} \cup \gamma_N(c)(s \setminus (c,2)), & (c \in C_N). \end{cases}$$

[3] It appears that our condition only requires that the cell c be filled somehow in $\gamma_M(c)(x)$; however, monotonicity of σ ensures that if it is filled in $\gamma_M(c)(s)$, then it must be filled *with the same value* in s.

- **Quantifiers** $Q_\alpha.\varphi$. Let $M = [\![\varphi]\!]$.

$$\begin{array}{lcl} \gamma_{Q_\alpha(M)}(c_0)(s) & = & \varnothing, \\ \gamma_{Q_\alpha(M)}(c)(\{(c_0, a)\} \uplus s) & = & \{(c_0, a)\} \cup \gamma_M(c)(s) \quad (c \in C_M). \end{array}$$

  Thus the choice initially made by α to decide the value of the quantifier is visible to all the agents.

- **Parallel Composition** $\varphi \| \psi$. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$. We define $\gamma_{M\|N}$ by:

$$\gamma_{M\|N}(c)(s) = \begin{cases} \gamma_M(c)(\pi_M(s)), & c \in C_M \\ \gamma_N(c)(\pi_N(s)), & c \in C_N. \end{cases}$$

  Here π_M, π_N are the *projection functions*; e.g.

$$\pi_M(s) = \{(c, v) \in s \mid c \in C_M\}.$$

  Thus the view at a cell in the sub-game M or N is what it would have been if we were playing only in that sub-game. This implements a complete block on information flow between the two sub-games. It can be seen as corresponding directly to the Linear Logic connective $\otimes$.

- **Sequential Composition** $\varphi \cdot \psi$. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$. We define $\gamma_{M \cdot N}$ by:

$$\gamma_{M \cdot N}(c)(s) = \begin{cases} \gamma_M(c)(\pi_M(s)), & c \in C_M \\ \pi_M(s) \cup \gamma_N(c)(\pi_N(s)), & c \in C_N. \end{cases}$$

  Thus while we are playing in M, visibility is at it was in that sub-game. When we have finished a complete play s in M and start to play in N, we can see the whole completed play s, together with what is visible in the sub-game N.

- **Role Permutation** $\hat{\pi}(\varphi)$. We set $\gamma_{[\![\hat{\pi}(\varphi)]\!]} = \gamma_{[\![\varphi]\!]}$. The same information is available from each cell; but, for example, if agent α had more information available than agent β in M, that advantage will be transferred to β in $\hat{\pi}(M)$ if π interchanges α and β.

3.2.4 Global definitions of strategy sets

We define the strategy sets $S_\alpha([\![\varphi]\!])$ compositionally from the construction of φ. The main point to note is the constructions on strategies which arise in making these definitions; these show the *functorial character* of the game-semantical interpretation of the connectives, and point the way towards a *semantics of proofs*—or indeed, in the first instance, to what the proof system should be—for the logic.

- **Atomic formulas and constant 1**. These cases are trivial, since the set of cells is empty. Thus $\mathcal{D}(\llbracket A \rrbracket) = \mathcal{D}(\llbracket \mathbf{1} \rrbracket) = \{\varnothing\}$. We set $S_\alpha(\llbracket A \rrbracket) = S_\alpha(\llbracket \mathbf{1} \rrbracket) = \{\mathsf{id}_{\{\varnothing\}}\}$.
- **Choice connectives** $\varphi \oplus_\alpha \psi$. Let $M = \llbracket \varphi \rrbracket$, $N = \llbracket \psi \rrbracket$. We firstly define some constructions on closure operators:

$$\mathsf{in}_1 : \mathsf{Cl}_\alpha(M) \longrightarrow \mathsf{Cl}_\alpha(M \oplus_\alpha N), \quad \mathsf{in}_2 : \mathsf{Cl}_\alpha(N) \longrightarrow \mathsf{Cl}_\alpha(M \oplus_\alpha N)$$

$$\oplus : \mathsf{Cl}_\beta(M) \times \mathsf{Cl}_\beta(N) \longrightarrow \mathsf{Cl}_\beta(M \oplus_\alpha N) \ \ (\beta \neq \alpha).$$

For $i = 1, 2$:

$$\mathsf{in}_i(\sigma)(s) = \begin{cases} \{(c_0, i)\} \cup \sigma(s \setminus \{(c_0, i)\}), & (c_0, j) \in s \ \Rightarrow \ j = i \\ \top & \text{otherwise.} \end{cases}$$

$$\begin{array}{rcl} \sigma \oplus \tau(\varnothing) & = & \varnothing \\ \sigma \oplus \tau(\{(c_0, 1)\} \uplus s) & = & \{(c_0, 1)\} \cup \sigma(s) \\ \sigma \oplus \tau(\{(c_0, 2)\} \uplus t) & = & \{(c_0, 2)\} \cup \tau(t). \end{array}$$

Thus $\mathsf{in}_1(\sigma)$ is the strategy for α in $M \oplus_\alpha N$ which firstly decides to play in M, and then subsequently plays like σ, which is ("inductively") assumed to be a strategy for α in M. Similarly for $\mathsf{in}_2(\tau)$. Note that both these strategies are "non-strict": that is, $\mathsf{in}_i(\sigma)(\varnothing)$ will at least contain (c_0, i). This reflects the idea that agent α must play the first move in $M \oplus_\alpha N$, and nothing can happen until he does. The strategy $\sigma \oplus \tau$ for another agent $\beta \neq \alpha$ must on the other hand "wait" at the initial state $\varnothing$ until α has made its decision. Once this has happened, it plays according to σ if the decision was to play in M, and according to τ if the decision was to play in N.

Note how the definitions of in_1, in_2 show why strategies must in general be partial; there is a "self-consistency" condition that we should only be confronted with situations in which the decisions ascribed to us are indeed those we actually made.

We now define the strategy sets for $M \oplus_\alpha N$:

$$\begin{array}{rcl} S_\alpha(M \oplus_\alpha N) & = & \{\mathsf{in}_1(\sigma) \mid \sigma \in S_\alpha(M)\} \cup \{\mathsf{in}_2(\tau) \mid \tau \in S_\alpha(N)\} \\ S_\beta(M \oplus_\alpha N) & = & \{\sigma \oplus \tau \mid \sigma \in S_\beta(M) \ \wedge \ \tau \in S_\beta(N)\} \quad (\beta \neq \alpha). \end{array}$$

- **Quantifiers** $Q_\alpha(\varphi)$. Let $M = [\![\varphi]\!]$. We define operations

$$\oplus_{a\in\mathcal{I}} : \mathsf{Cl}_\beta(M)^{\mathcal{I}} \longrightarrow \mathsf{Cl}_\beta(Q_\alpha(M)), \quad (\beta \neq \alpha),$$

and, for each $a \in \mathcal{I}$:

$$\mathsf{up}_a : \mathsf{Cl}_\alpha(M) \longrightarrow \mathsf{Cl}_\alpha(Q_\alpha(M)).$$

$$\mathsf{up}_a(\sigma)(s) = \begin{cases} \{(c_0, a)\} \cup \sigma(s \setminus \{(c_0, a)\}), & (c_0, b) \in s \ \Rightarrow\ b = a \\ \top & \text{otherwise.} \end{cases}$$

$$\begin{array}{lcl} (\oplus_{a\in\mathcal{I}}\sigma_a)(\varnothing) & = & \varnothing \\ (\oplus_{a\in\mathcal{I}}\sigma_a)(\{(c_0, b)\} \uplus s) & = & \{(c_0, b)\} \cup \sigma_b(s). \end{array}$$

Note the similarity of these operations to those defined for the choice connectives. (In fact, the separated sum $M \oplus N$ can be seen as the composite $(M + N)_\perp$ of disjoint sum and lifting constructions.) Note also, in the definition of $\oplus_{a\in\mathcal{I}}\sigma_a$, the dependence of the strategy σ_b used to continue the play on the element $b \in \mathcal{I}$ initially chosen by α.

We define the strategy sets as follows:

$$\begin{array}{lcl} S_\alpha(Q_\alpha(M)) & = & \{\mathsf{up}_a(\sigma) \mid a \in \mathcal{I}, \sigma \in S_\alpha(M)\} \\ S_\beta(Q_\alpha(M)) & = & \{\oplus_{a\in\mathcal{I}}\sigma_a \mid \forall a \in \mathcal{I}.\, \sigma_a \in S_\beta(M)\} \quad (\beta \neq \alpha). \end{array}$$

- **Parallel Composition** $\varphi\|\psi$. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$. We define an operation

$$\| : \mathsf{Cl}_\alpha(M) \times \mathsf{Cl}_\alpha(N) \longrightarrow \mathsf{Cl}_\alpha(M\|N)$$

$$\sigma\|\tau(s) = \sigma(\pi_M(s)) \cup \tau(\pi_N(s)).$$

This is just the functorial action of the product, and gives the "information independence" of play in the two sub-games. Now we define the strategy sets:

$$S_\alpha(M\|N) = \{\sigma\|\tau \mid \sigma \in S_\alpha(M) \ \wedge\ \tau \in S_\alpha(N)\}.$$

- **Sequential Composition** $\varphi \cdot \psi$. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$. Given $\sigma \in \mathsf{Cl}_\alpha(M)$, and a family $(\tau_s)_{s\in\mathsf{Max}(M)} \subseteq \mathsf{Cl}_\alpha(N)$ indexed by maximal states in M, we define:

$$(\sigma \cdot (\tau_s)_s)(t) = \begin{cases} \sigma(t), & t \in \mathcal{D}(M), \sigma(t) \notin \mathsf{Max}(M) \\ \sigma(t) \cup \tau_{\sigma(t)}(\varnothing), & t \in \mathcal{D}(M), \sigma(t) \in \mathsf{Max}(M) \\ s \cup \tau_s(u), & t = s \cup u, s \in \mathsf{Max}(M), u \in \mathcal{D}(N). \end{cases}$$

This allows for arbitrary dependency of agent α's play in N on what previously occurred in M. Note that in the case that σ completes a maximal play s in M, control passes immediately to τ_s to continue in N. We then define

$$S_\alpha(M \cdot N) = \{\sigma \cdot (\tau_s)_s \mid \sigma \in S_\alpha(M) \ \wedge \ \forall s \in \mathsf{Max}(M).[\tau_s \in S_\alpha(N)]\}.$$

- **Role Permutation** $\hat{\pi}(\varphi)$. Let $M = [\![\varphi]\!]$. Here we simply set $S_\alpha(\hat{\pi}(M)) = S_{\pi^{-1}(\alpha)}(M)$.

3.2.5 Comparison of the local and global definitions

Having defined strategy sets in these two contrasting fashions, we must compare them. Let φ be a formula of $\mathcal{L}_\mathcal{A}$. We write $S^{\mathsf{g}}_\alpha([\![\varphi]\!])$ for the strategy set for $[\![\varphi]\!]$ defined according to the global construction, and similarly $S^{\mathsf{l}}_\alpha([\![\varphi]\!])$ for the local definition (1.2) using the visibility function $\gamma_{[\![\varphi]\!]}$.

Proposition 3.2. For all $\varphi \in \mathcal{L}_\mathcal{A}$, $S^{\mathsf{g}}_\alpha([\![\varphi]\!]) \subseteq S^{\mathsf{l}}_\alpha([\![\varphi]\!])$.

Proof. A straightforward induction on φ. Note in particular that $\sigma \| \tau$ satisfies the local condition (1.2) with respect to the parallel composition.

Q.E.D.

The converse is false in general. The strategies in $S^{\mathsf{g}}_\alpha([\![\varphi]\!])$ have two important global properties which strategies satisfying the local condition (1.2) need not possess.

Safety We define the *domain* $\mathsf{dom}(\sigma)$ of a closure operator $\sigma \in \mathsf{Cl}_\alpha(M)$ to be the least subset of $\mathcal{D}(M)^\top$ satisfying the following conditions:

$$\begin{array}{ll} (D1) & \varnothing \in \mathsf{dom}(\sigma) \\ (D2) & s \in \mathsf{dom}(\sigma) \ \Rightarrow \ \sigma(s) \in \mathsf{dom}(\sigma) \\ (D3) & S \subseteq \mathsf{dom}(\sigma), \ S \text{ directed} \ \Rightarrow \ \bigcup S \in \mathsf{dom}(\sigma) \\ (D4) & s \in \mathsf{dom}(\sigma), s \subseteq t \in \mathcal{D}(M), \\ & \qquad [(c,v) \in t \setminus s \ \Rightarrow \ \lambda_M(c) \neq \alpha] \ \Rightarrow \ t \in \mathsf{dom}(\sigma). \end{array}$$

Note that $(D1)$–$(D3)$ are the usual inductive definition of the set of iterations leading to the least fixpoint of σ. The condition $(D4)$ gives this definition its game-theoretic or multi-agent character.

We say that σ is *safe* if $\top \notin \mathsf{dom}(\sigma)$. Thus if σ is never confronted by α-moves that it would not itself have made, then whatever the other agents do it will not "crash" or "abort" (which is how we think of $\top$).

Proposition 3.3. For all $\varphi \in \mathcal{L}_\mathcal{A}$, every strategy in $S^{\mathsf{g}}_\alpha([\![\varphi]\!])$ is safe.

Progress We consider the following assumptions on strategies $\sigma \in \mathsf{Cl}_\alpha(M)$:

- (WP) If $s \in \mathcal{D}(M)$ contains an enabling of some α-cell c which is not filled in s, then $\sigma(s) \neq s$. In other words, σ does *something* (makes at least one decision) whenever it can.
- (MP) For all $s \in \mathcal{D}(M)$, if $\sigma(s)$ contains an enabling of some α-cell c, then it fills c. Thus σ decides *every* α-cell as soon as it becomes accessible.

We call (WP) the *weak progress assumption* and (MP) the *maximal progress assumption.* Clearly (MP) implies (WP).

Lemma 3.4. The weak progress assumption implies the maximal progress assumption, and hence the two conditions are equivalent.

Proof. Let σ be an α-strategy not satisfying (MP). Then there must be a state s such that some α-cells are accessible but not filled in $\sigma(s)$. By idempotence, $\sigma(\sigma(s)) = \sigma(s)$, and hence σ does not satisfy (WP). Q.E.D.

Proposition 3.5. For all $\varphi \in \mathcal{L}_\mathcal{A}$, every strategy in $S^{\mathsf{g}}_\alpha(\llbracket\varphi\rrbracket)$ satisfies the maximal progress assumption (MP).

Given an $\mathcal{A}$-game M, we define $S^{\mathsf{lsp}}_\alpha(M)$ to be the set of all strategies in $S^{\mathsf{l}}_\alpha(M)$ which are safe and satisfy the weak progress assumption.

Theorem 3.6 (Characterization Theorem)**.** For all $\varphi \in \mathcal{L}_\mathcal{A}$, $S^{\mathsf{g}}_\alpha(\llbracket\varphi\rrbracket) = S^{\mathsf{lsp}}_\alpha(\llbracket\varphi\rrbracket)$.

Proof. By induction on φ. We indicate some cases.

1. Parallel composition. For a strategy $\sigma \in S^{\mathsf{lsp}}_\alpha(M\|N)$, the visibility condition implies that $\sigma = \sigma_1\|\sigma_2$. The safety of σ implies that of σ_1 and σ_2, and similarly (MP) for σ implies (MP) for σ_1 and σ_2. Thus $\sigma_1 \in S^{\mathsf{lsp}}_\alpha(M)$ and $\sigma_2 \in S^{\mathsf{lsp}}_\alpha(N)$, and we can apply the induction hypothesis. The case for sequential composition is similar.
2. Choice connectives. Here the progress assumption implies that the strategy holding the initial cell must fill it. Safety implies that strategies for other players must have the form $\sigma \oplus \tau$. Play after the initial cell is filled reduces to play in the chosen sub-game, and we can apply the induction hypothesis.

Q.E.D.

Thus we explicitly characterize the "immanent" properties of the strategy sets $S^{\mathsf{g}}_\alpha(\llbracket\varphi\rrbracket)$ in terms of local conditions on information visibility, plus safety and liveness properties.

3.3 Evaluation of strategy profiles

Consider a CDS M with a strategy set S_α for each agent $\alpha \in \mathcal{A}$. A *strategy profile* is an $\mathcal{A}$-tuple

$$(\sigma_\alpha)_{\alpha \in \mathcal{A}} \in \prod_{\alpha \in \mathcal{A}} S_\alpha$$

which picks out a choice of strategy for each $\alpha \in \mathcal{A}$. The key operation in "bringing the semantics to life" is to define the result or *outcome* of playing these strategies off against each other. Given the concurrent nature of our games, and the complex forms of temporal dependency and information flow which may arise in them, it might seem that a formal definition of this operation will necessarily be highly complex and rather messy. In fact, our mathematical framework allows for a very elegant and clean definition. We shall use the notation $\langle \sigma_\alpha \rangle_{\alpha \in \mathcal{A}}$ for this operation. It maps strategy profiles to *states of M*. The idea is that the state arising from $\langle \sigma_\alpha \rangle_{\alpha \in \mathcal{A}}$ will be that reached by starting in the initial state $\varnothing$, and repeatedly playing the strategies in the profile until no further moves can be made. The formal definition is as follows.

Definition 3.7. We define $\langle \sigma_\alpha \rangle_{\alpha \in \mathcal{A}}$ to be the *least common fixpoint* of the family of closure operators $(\sigma_\alpha)_{\alpha \in \mathcal{A}}$; *i.e.* the least element s of $\mathcal{D}(M)^\top$ such that $\sigma_\alpha(s) = s$ for all $\alpha \in \mathcal{A}$.

The following Proposition (which is standard) guarantees that this definition makes sense.

Proposition 3.8. Any family of closure operators C on a complete lattice L has a common least fixpoint. In case the lattice has finite height, or $C = \{c_1, \ldots, c_n\}$ is finite and the closure operators in C are continuous (*i.e.* preserve directed joins) this common least fixpoint can be obtained constructively by the expression

$$\bigvee_{k \in \omega} c^k(\bot), \text{ where } c = c_1 \circ \cdots \circ c_k.$$

Any permutation of the order of the composition in defining c, or indeed any "schedule" which ensures that each closure operator is applied infinitely often, will lead to the same result.

We recall the notion of *safety* from the previous section.

Proposition 3.9. Let $(\sigma_\alpha)_{\alpha \in \mathcal{A}}$ be a strategy profile in which σ_α is safe for all $\alpha \in \mathcal{A}$. Then $\langle \sigma_\alpha \rangle_{\alpha \in \mathcal{A}} \neq \top$.

Proof. We prove by transfinite induction on the iterations towards the least fixpoint that every state which is reached is in the domain of every strategy in the profile. The base case is $(D1)$, and the limit ordinal case is $(D3)$. Applying some σ_α to the current state stays in the domain of σ_α by $(D2)$, and in the domain of every other strategy in the profile by $(D4)$. Q.E.D.

In particular, we know by Proposition 3.3 that this applies to our setting, where we have a formula $\varphi \in \mathcal{L}_{\mathcal{A}}$, with corresponding CDS $M = [\![\varphi]\!]$ and strategy sets $S_\alpha([\![\varphi]\!])$, $\alpha \in \mathcal{A}$. Furthermore, we have:

Proposition 3.10. For all $\varphi \in \mathcal{L}_{\mathcal{A}}$, and every strategy profile $(\sigma_\alpha)_{\alpha\in\mathcal{A}} \in S^{\mathsf{g}}_\alpha([\![\varphi]\!]) = S^{\mathsf{lsp}}_\alpha([\![\varphi]\!])$:

$$\langle\sigma_\alpha\rangle_{\alpha\in\mathcal{A}} \in \mathsf{Max}([\![\varphi]\!]).$$

Proof. Firstly, we know by Proposition 3.9 that $\langle\sigma_\alpha\rangle_{\alpha\in\mathcal{A}} \neq \top$. Let $s = \langle\sigma_\alpha\rangle_{\alpha\in\mathcal{A}}$. If s is not maximal, some cell c must be accessible but not filled in s. Suppose c is an α-cell. Since σ_α satisfies (WP), we must have $\sigma_\alpha(s) \neq s$, contradicting the definition of $\langle\sigma_\alpha\rangle_{\alpha\in\mathcal{A}}$ as a common fixpoint of all the strategies in the profile. Q.E.D.

Thus the outcome of evaluating a strategy profile in the game arising from any formula is always a well-defined maximal state.

We pause to mention another connection with Theoretical Computer Science. Our use of closure operators as strategies, and the definition of the evaluation of strategy profiles as least common fixpoints, builds on ideas which arose originally in the semantics of *dataflow* [Ja$_1$Pa$_2$Pi$_2$89] and *concurrent constraint programming* [Sa$_5$Ri$_0$Pa$_2$91]. They have also been applied extensively to constraint programming and constraint propagation algorithms over the past decade [Ap97]. Our own previous development of these ideas appears in a number of papers [AbMe$_1$99, Ab00a, Ab03].

3.4 Outcomes and valuations

One ingredient which has been missing thus far in our account of the semantics of $\mathcal{L}_{\mathcal{A}}$ has been any notion of *payoff* or *utility* in game-theoretic terms, or of *truth-valuation* in logical terms, which may serve as a basis for game-theoretical notions of *equilibria* or logical notions such as *validity.* The status of the standard notions on the logical side is far from clear when we pass to multi-agent games. However, we can certainly provide support for studying equilibrium notions in our framework, in such a way that these specialize to the usual logical notions in the two-agent case. We shall only enter into a preliminary discussion here, simply to indicate some of the possibilities.

As we have just seen, for the games arising from formulas in $\mathcal{L}_{\mathcal{A}}$, evaluation of strategy profiles always leads to maximal states. Moreover, the CDS corresponding to any formula has only finitely many *cells* (although if

$\mathcal{I}$ is infinite, so also will be the sets of values, decisions and states). Hence any state consists of only finitely many decisions.

Proposition 3.11. For any closed formula $\varphi \in \mathcal{L}_{\mathcal{A}}$, a maximal state in $[\![\varphi]\!]$ corresponds to a combination of atomic sentences, built by series and parallel composition from (instances of) atomic subformulas of φ. If φ is built from atomic formulas using only the choice connectives and quantifiers, maximal states will correspond exactly to single instances of atomic subformulas.

Proof. Given a maximal state s, we argue by induction on φ. We indicate some cases:

- $\varphi \oplus_\alpha \psi$. Then s contains (c_0, i). If $i = 1$, we continue with the formula φ and the maximal state of $[\![\varphi]\!]$ obtained by removing $(c_0, 1)$ from s. Similarly if $i = 2$.
- $Q_\alpha x.\, \varphi$. Then s contains (c_0, a). We continue inductively with $\varphi[a/x]$ and the maximal state of $[\![\varphi[a/x]]\!]$ obtained by removing (c_0, a) from s.
- $\varphi \| \psi$. We continue inductively with φ and $\pi_M(s)$, and with ψ and $\pi_N(s)$, and glue the results back together with parallel composition.
- $\varphi \cdot \psi$. Essentially the same as for parallel composition.

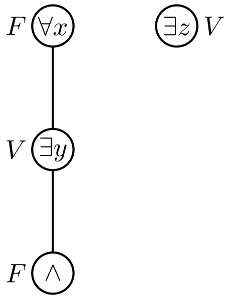
Q.E.D.

3.4.1 Example

We take $\mathcal{A} = \{V, F\}$, and use standard notation for choice connectives and quantifiers. Consider the formula

$$\forall x.\exists y.[A(x,y) \ \wedge \ B(y)] \parallel \exists z.C(z).$$

The corresponding CDS has four cells:



In a maximal state of this CDS, these cells are all filled. If the $\forall x$ cell is filled with $a \in \mathcal{I}$, $\exists y$ with $b \in \mathcal{I}$, $\wedge$ with 1, and $\exists z$ with $c \in \mathcal{I}$, then the state will correspond to the following parallel composition of atomic sentences:

$$A(a, b) \parallel C(c).$$

In the usual Hintikka-style Game semantics, a model $\mathcal{M}$ is used to evaluate atomic sentences. We can see that in our setting, this work can equivalently be done by a *Boolean valuation function*

$$\mathsf{val} : \mathsf{Max}([\![\varphi]\!]) \longrightarrow \{0, 1\}.$$

So for 2-agent games, we could simply use such a valuation to give a notion of winning strategy for Verifier, and hence of logical validity.

More generally, in the multi-agent case we should consider valuations

$$\mathsf{val}_\alpha : \mathsf{Max}([\![\varphi]\!]) \longrightarrow \mathcal{V}_\alpha$$

for each agent α, into some set of *preferences* or *utilities*. Given an outcome

$$o = \langle \sigma_\alpha \rangle_{\alpha \in \mathcal{A}} \in \mathsf{Max}([\![\varphi]\!]),$$

we can evaluate it from the perspective of each agent α as $\mathsf{val}_\alpha(o)$, and hence formulate notions such as Nash equilibrium and other central game-theoretical notions.

3.4.2 Compositionality?

Until now our semantics has been fully compositional. However, as things stand, valuation functions *cannot* be described in a compositional fashion. The problem becomes clear if we consider our treatment of atomic formulas. Their current representation in our semantics is vacuous — the empty CDS. This carries no information which can be used by a valuation function to express the dependence of an outcome on the values previously chosen for the variables appearing in the atomic formula. We can recover this correspondence globally, as in Proposition 3.11, but not *compositionally*, by gluing together a valuation function defined for an atomic formula with those arising from the context in which it occurs.

We shall now give a reformulation of the syntax of $\mathcal{L}_\mathcal{A}$ which will allow us to take full account of the role of variables and variable-binding in our semantics, and hence provide the basis for a compositional treatment both of valuation functions, and of IF-quantifiers and other partial-information constructs.

4 Towards environmentally friendly logic

It is, or should be, an aphorism of semantics that:

The key to compositionality is parameterization.

Choosing the parameters aright allows the meaning of expressions to be made sensitive to their contexts, and hence defined compositionally. While this principle could—in theory—be carried to the point of trivialization, in practice the identification of the right form of parameterization does usually represent some genuine insight into the structure at hand.

We shall now describe an approach to making the syntax of quantifier particles, including IF-quantifiers, fully compositional. This can then serve as a basis for a fully compositional account of valuations on outcomes.

4.1 Syntax as a category

Note firstly a certain kind of quasi-duality between quantifiers and atomic formulas. Quantifiers $Q_\alpha x$ project the scope of x inwards over sequential compositions (but not across parallel compositions). Atomic formulas $A(x_1, \ldots, x_n)$ depend on variables coming from an outer scope.

Now consider IF-quantifiers $\forall x/y$ which bind x, but also declare that it does *not* depend on an outer quantification over y. This is a peculiar binding construct, quite apart from its semantic interpretation. The bidirectional reach of the scope—inwards for x, outwards for y—is unusual, and difficult to make sense of in isolation from a given context of use. So in fact, it seems hard to give a decent compositional *syntax* for IF-quantifiers, before we even start to think about semantics.

Once again, there is work coming from Theoretical Computer Science which is suggestive: namely the π-calculus [MiPa$_6$Wa$_1$92a, MiPa$_6$Wa$_1$92b], with its *scope restriction* and *extrusion*. The action calculi subsequently developed by Milner [Mi93] are even more suggestive, although only certain features are relevant here.

We shall reformulate our view of logical syntax as follows. Each syntactic constituent will have an *arity* and a *co-arity*. Concretely, we shall take these arities and co-arities to be finite sets of variables, although algebraically we could just take them to be natural numbers. We shall write a syntactic expression as

$$\varphi : X \longrightarrow Y$$

where X is the arity, and Y is the co-arity. The idea is that the arity specifies the variables that φ expects to have bound by its outer environment, while the co-arity represents variables that it is binding with respect to its inner environment.

The quantifier particle $Q_\alpha x$ can be described in these terms as

$$Q_\alpha x : \varnothing \longrightarrow \{x\} \tag{1.3}$$

or more generally as

$$Q_\alpha x : X \longrightarrow X \uplus \{x\}.$$

An atom $A(x_1, \ldots, x_n)$ will have the form

$$A(x_1, \ldots, x_n) : \{x_1, \ldots, x_n\} \longrightarrow \varnothing,$$

so we indeed see a duality with (1.3).

We specify "typed" versions of sequential and parallel composition with respect to these arities and co-arities:

$$\frac{\varphi : X \longrightarrow Y \quad \psi : Y \longrightarrow Z}{\varphi \cdot \psi : X \longrightarrow Z} \qquad \frac{\varphi : X_1 \longrightarrow Y_1 \quad \psi : X_2 \longrightarrow Y_2}{\varphi \| \psi : X_1 \uplus X_2 \longrightarrow Y_1 \uplus Y_2}.$$

The constant **1** has the form

$$\mathbf{1} : X \longrightarrow \varnothing$$

for any X.

We take these syntactic expressions modulo a notion of *structural congruence*, as in the π-calculus and action calculi. We impose the axioms

$$\varphi \cdot (\psi \cdot \theta) \equiv (\varphi \cdot \psi) \cdot \theta, \qquad \mathbf{1} \cdot \varphi \equiv \varphi \equiv \varphi \cdot \mathbf{1}$$

wherever these expressions make sense with respect to the typing with arities and co-arities.

Thus we are in fact describing a *category* $\mathbf{C}(\mathcal{L}_\mathcal{A})$. The objects are the arities—"co-arities" are simply arities appearing as the codomains of arrows in the category. The arrows are the syntactic expressions modulo structural congruence; and the composition in the category is sequential composition.

To complete the picture: for the choice connectives, we have

$$\frac{\varphi : X \longrightarrow \varnothing \quad \psi : X \longrightarrow \varnothing}{\varphi \oplus_\alpha \psi : X \longrightarrow \varnothing}$$

and for role interchange

$$\frac{\varphi : X \longrightarrow Y}{\hat{\pi}(\varphi) : X \longrightarrow Y}.$$

For the IF-quantifier we have

$$\forall x/y : X \uplus \{y\} \longrightarrow X \uplus \{x\} \uplus \{y\},$$

which makes explicit the fact that y occurs free in $\forall x/y$.

The arrows in $\mathbf{C}(\mathcal{L}_\mathcal{A})$ will be the well-formed formulas (both open and "co-open") of our logic. In particular, the sentences or closed formulas will be the arrows of the form $\varphi : \varnothing \longrightarrow \varnothing$.

4.2 Static semantics revisited

We consider the static semantics of a syntactic constituent $\varphi : X \to Y$. The $\mathcal{A}$-game $[\![\varphi]\!]$ defined as in Section 3.1 remains unchanged. In particular, atomic formulas are still assigned the empty $\mathcal{A}$-game. The new ingredient in the static semantics will reflect the intentions behind the arities and coarities, which we now set out in greater detail. The arity X is the set of variables being imported (as "free variables") from the outer environment by φ. Thus an "open formula" in the usual sense will be an arrow of type $X \to \varnothing$. The novel feature in our approach to logical syntax, following Milner's action calculi, are the co-arities. In $\varphi : X \to Y$, it is useful to write

$$Y = (X \cap Y) \uplus (Y \setminus X).$$

Now:

- $X \cap Y$ represents those variables imported from the outer environment which we simply "pass on through" to be imported in turn by the inner environment. (The variables in $X \setminus Y$ are hidden from the inner environment.)
- $Y \setminus X$ represents the variables which are being *defined* by φ, and exported to the inner environment, where they will bind free occurrences of those variables.

As we have seen, variables bound by quantifiers are interpreted in our semantics by cells where localized decisions can be made. Hence the act of defining a variable amounts to binding it to a cell. Thus the single new component in the static semantics of $\varphi : X \to Y$ will be a function

$$\mathsf{bind}_M : Y \setminus X \longrightarrow C_M$$

where $M = [\![\varphi]\!]$. We now show how bind is defined compositionally.

- **Atomic formulas, constant 1, choice connectives** $\oplus_\alpha$. These cases are all trivial, since the types are of the form $X \to \varnothing$, and $\varnothing \setminus X = \varnothing$.
- **Quantifiers** $Q_\alpha x : X \to X \uplus \{x\}$. As we saw in Section 3.1, $[\![Q_\alpha x]\!]$ has a single cell c_0. We set

$$\mathsf{bind}_{[\![Q_\alpha x]\!]}(x) = c_0.$$

- **Parallel Composition** $\varphi \| \psi : X_1 \uplus X_2 \longrightarrow Y_1 \uplus Y_2$, where $\varphi : X_1 \to Y_1$, $\psi : X_2 \to Y_2$. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$. We define

$$\mathsf{bind}_{M\|N}(y) = \begin{cases} \mathsf{bind}_M(y), & y \in Y_1 \setminus (X_1 \cup X_2) \\ \mathsf{bind}_N(y), & y \in Y_2 \setminus (X_1 \cup X_2). \end{cases}$$

- **Sequential Composition** $\varphi \cdot \psi : X \rightarrow Z$, where $\varphi : X \rightarrow Y$ and $\psi : Y \rightarrow Z$. This is the key case. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$. We can write
$$Z \setminus X \;\; = \;\; (Z \setminus (X \cup Y)) \uplus (Z \cap (Y \setminus X)).$$
Hence we can define:
$$\mathsf{bind}_{M \cdot N}(z) = \begin{cases} \mathsf{bind}_M(z), & z \in Z \cap (Y \setminus X) \\ \mathsf{bind}_N(z), & z \in Z \setminus (X \cup Y). \end{cases}$$
- **Role Interchange** $\hat{\pi}(\varphi)$. Let $M = [\![\varphi]\!]$.
$$\mathsf{bind}_{\hat{\pi}(M)} = \mathsf{bind}_M.$$

4.3 Interlude: structural congruence

We have already introduced a notion of structural congruence $\equiv$ in defining the syntactic category $\mathbf{C}(\mathcal{L}_{\mathcal{A}})$. We now consider the interpretation of the structural congruence using the static semantics, and the issue of axiomatization.

We say that our semantics validates a structural congruence
$$\varphi \equiv \psi : X \longrightarrow Y$$
if $[\![\varphi]\!] \cong [\![\psi]\!]$, that is if the $\mathcal{A}$-games they denote are isomorphic (in the usual sense of isomorphism for relational structures).

Proposition 4.1. *The following axioms for structural congruence are valid in the static semantics:*
$$\begin{array}{rcl}
\varphi \cdot (\psi \cdot \theta) & \equiv & (\varphi \cdot \psi) \cdot \theta \\
\mathbf{1} \cdot \varphi & \equiv & \varphi \\
\varphi & \equiv & \varphi \cdot \mathbf{1} \\
\varphi \| (\psi \| \theta) & \equiv & (\varphi \| \psi) \| \theta \\
\mathbf{1} \| \varphi & \equiv & \varphi \\
\varphi & \equiv & \varphi \| \mathbf{1} \\
\varphi \| \psi & \equiv & \psi \| \varphi \\
\varphi \oplus_\alpha \psi & \equiv & \psi \oplus_\alpha \varphi \\
\hat{\pi}_1(\hat{\pi}_2(\varphi)) & \equiv & \widehat{\pi_1 \circ \pi_2}(\varphi) \\
\hat{\pi}(\varphi \oplus_\alpha \psi) & \equiv & \hat{\pi}(\varphi) \oplus_{\pi(\alpha)} \hat{\pi}(\psi) \\
\hat{\pi}(Q_\alpha x) & \equiv & Q_{\pi(\alpha)} x
\end{array}$$

$$\begin{aligned}\hat{\pi}(\varphi \cdot \psi) &\equiv \hat{\pi}(\varphi) \cdot \hat{\pi}(\psi) \\ \hat{\pi}(\varphi \| \psi) &\equiv \hat{\pi}(\varphi) \| \hat{\pi}(\psi).\end{aligned}$$

Remark 4.2. The following are in general *not* valid in the static semantics (which we write colourfully if imprecisely as $\not\equiv$):

$$\begin{aligned}\varphi \oplus_\alpha \varphi &\not\equiv \varphi \\ \varphi \oplus_\alpha \mathbf{1} &\not\equiv \varphi \\ \varphi \oplus_\alpha (\psi \oplus_\alpha \theta) &\not\equiv (\varphi \oplus_\alpha \psi) \oplus_\alpha \theta \\ (\varphi \oplus_\alpha \psi) \cdot \theta &\not\equiv (\varphi \cdot \theta) \oplus_\alpha (\psi \cdot \theta) \\ \varphi \cdot (\psi \oplus_\alpha \theta) &\not\equiv (\varphi \cdot \psi) \oplus_\alpha (\varphi \cdot \theta) \\ \varphi \| (\psi \oplus_\alpha \theta) &\not\equiv (\varphi \| \psi) \oplus_\alpha (\varphi \| \theta) \\ (\varphi_1 \| \psi_1) \cdot (\varphi_2 \| \psi_2) &\not\equiv (\varphi_1 \cdot \varphi_2) \| (\psi_1 \cdot \psi_2).\end{aligned}$$

Remark 4.3. If we weaken the notion of validity of $\varphi \equiv \psi$ to $\mathcal{D}([\![\varphi]\!]) \cong \mathcal{D}([\![\varphi]\!])$ (order-isomorphism), then the only one of the above non-equivalences which becomes valid is

$$(\varphi \oplus_\alpha \psi) \cdot \theta \equiv (\varphi \cdot \theta) \oplus_\alpha (\psi \cdot \theta).$$

Conjecture 4.4. The axioms listed in Proposition 4.1 are complete for validity in the static semantics.

4.4 Valuations

We are now in a position to give a compositional definition of valuation functions on outcomes. For each agent $\alpha \in \mathcal{A}$ we shall fix a set $\mathcal{V}_\alpha$ of values (utilities, payoffs, truth-values ...). What structure should $\mathcal{V}_\alpha$ have? In order to express preferences between outcomes, and hence to capture the classical game-theoretic solution concepts such as Nash equilibrium, we would want $\mathcal{V}_\alpha$ to carry an order structure. For our present purposes, we need only that $\mathcal{V}_\alpha$ carries two binary operations

$$\odot, \otimes : \mathcal{V}_\alpha^2 \longrightarrow \mathcal{V}_\alpha$$

and an element $1 \in \mathcal{V}_\alpha$, such that $(\mathcal{V}_\alpha, \odot, 1)$ is a monoid, and $(\mathcal{V}_\alpha, \otimes, 1)$ is a commutative monoid.[4]

[4] A plausible general suggestion is to identify these two algebraic structures, and to take $\mathcal{V}_\alpha$ to be a (commutative) *quantale*, *i.e.* a sup-lattice-enriched monoid. These structures have been used fairly extensively in Computer Science over the past 15 years [AbVi93, Ba$_4$Co$_0$Sa$_0$05].

Now let $M : X \to Y$ be an $\mathcal{A}$-game, e.g. $M = [\![\varphi]\!]$, for $\varphi : X \to Y$ in $\mathbf{C}(\mathcal{L}_{\mathcal{A}})$. Then for each $\alpha \in \mathcal{A}$, an α-valuation function will have the form

$$\mathsf{val}_{M,\alpha} : \mathcal{I}^X \times \mathsf{Max}(M) \longrightarrow \mathcal{V}_\alpha. \tag{1.4}$$

Note firstly that we are only considering valuations applied to *maximal* states, which is reasonable since by Proposition 3.10, these are the only possible outcomes of evaluating strategy profiles. However, in a more general setting where infinite plays are possible, e.g. in the games arising from fixpoint extensions of $\mathcal{L}_{\mathcal{A}}$, one should consider *continuous valuations* defined on the whole domain of states $\mathcal{D}(M)$.

The form of $\mathsf{val}_{M,\alpha}$ expresses the dependency of the valuation both on the values of the variables being imported from the environment, and on the final state of play. There are two extremal cases:

1. If $X = \varnothing$, then the valuation simply reduces to a function $\mathsf{Max}(M) \longrightarrow \mathcal{V}_\alpha$ on maximal states. In particular, this will be the case for closed formulas.
2. If $C_M = \varnothing$, the valuation reduces to a function $\mathcal{I}^X \longrightarrow \mathcal{V}_\alpha$. This is exactly the usual kind of function from assignments to variables to (truth)-values induced by an atomic formula evaluated in a first-order model $\mathcal{M}$.

By allowing a range of intermediate cases between these two extremes, we can give a compositional account which, starting with given assignments of the form (2) for atomic formulas, ends with valuations of the form (1) for sentences.

We now give the compositional definition of the valuation function. We use $\eta \in \mathcal{I}^X$ to range over assignments to variables.

- **Atomic formulas**. We take the valuation functions $\mathcal{I}^X \longrightarrow \mathcal{V}_\alpha$ as given. Thus atomic formulas are operationally void—no moves, no plays—but valuationally primitive—generating the entire valuation function of any complex formula. This seems exactly right.
- **Constant 1**. We set $\mathsf{val}_{[\![\mathbf{1}]\!],\alpha}(\eta, \varnothing) = 1$.
- **Choice connectives** $\varphi \oplus_\beta \psi : X \to \varnothing$. Let $M = [\![\varphi]\!]$, $N = [\![N]\!]$.

$$\begin{array}{rcl} \mathsf{val}_{M \oplus_\beta N,\alpha}(\eta, \{(c_0, 1)\} \uplus s) & = & \mathsf{val}_{M,\alpha}(\eta, s) \\ \mathsf{val}_{M \oplus_\beta N,\alpha}(\eta, \{(c_0, 2)\} \uplus s) & = & \mathsf{val}_{N,\alpha}(\eta, s). \end{array}$$

- **Quantifiers** $Q_\beta x : X \to X \uplus \{x\}$.

$$\mathsf{val}_{[\![Q_\beta x]\!],\alpha}(\eta, \{(c_0, a)\}) = 1.$$

- **Parallel Composition** $\varphi \| \psi : X_1 \uplus X_2 \to Y_1 \uplus Y_2$, where $\varphi : X_1 \to Y_1$, $\psi : X_2 \to Y_2$. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$. Note that $\eta \in \mathcal{I}^{X_1 \uplus X_2}$ can be written as $\eta = (\eta_1, \eta_2)$, where $\eta_1 \in \mathcal{I}^{X_1}$, $\eta_2 \in \mathcal{I}^{X_2}$.

$$\mathsf{val}_{M\|N,\alpha}(\eta, s) = \mathsf{val}_{M,\alpha}(\eta_1, \pi_M(s)) \otimes \mathsf{val}_{N,\alpha}(\eta_2, \pi_N(s)).$$

- **Sequential Composition** $\varphi \cdot \psi : X \to Z$, where $\varphi : X \to Y$ and $\psi : Y \to Z$. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$.

$$\mathsf{val}_{M\cdot N,\alpha}(\eta, s) = \mathsf{val}_{M,\alpha}(\eta, \pi_M(s)) \odot \mathsf{val}_{N,\alpha}(\eta', \pi_N(s))$$

where $\eta' \in \mathcal{I}^Y$ is defined as follows:

$$\eta'(y) = \begin{cases} \eta(y), & y \in X \\ s(\mathsf{bind}_M(y)), & y \in Y \setminus X. \end{cases}$$

This is the key case—the only one where the bind function is used.

- **Role Interchange** $\hat{\pi}(\varphi)$. Let $M = [\![\varphi]\!]$.

$$\mathsf{val}_{\hat{\pi}(M),\alpha} = \mathsf{val}_{M,\pi^{-1}(\alpha)}.$$

4.5 Dynamic semantics revisited

Given an $\mathcal{A}$-game $M : X \to Y$, an α-strategy is a family $(\sigma_\eta)_{\eta \in \mathcal{I}^X}$, where $\sigma_\eta \in \mathsf{Cl}_\alpha(M)$ for all η. The definition of evaluation of strategy profiles is simply carried over pointwise to these families, so that we get an outcome for each $\eta \in \mathcal{I}^X$. The global definition of the strategy sets $S_\alpha(M)$ can also be carried over pointwise in a straightforward fashion. However, the explicit dependence on the values assigned to free variables also creates some new possibilities, in particular for giving semantics to IF-quantifiers. This is best discussed in terms of the local definition of information constraints via visibility functions, to which we now turn.

4.6 Visibility functions, occlusion and IF-quantifiers

We recall that the visibility function for an $\mathcal{A}$-game $M : X \to Y$ has the form

$$\gamma_M : C_M \longrightarrow [\mathcal{D}(M) \longrightarrow \mathcal{D}(M)]$$

and assigns a co-closure operator to each cell, specifying the information which is visible in any state to the agent wishing to fill the cell. We now augment this with an assignment

$$\mathsf{Occ}_M : C_M \longrightarrow \mathcal{P}(X).$$

The idea is that $\mathsf{Occ}_M(c) \subseteq X$ is the set of variables which are *occluded* at the cell c; hence the decision made at c cannot depend on the values of these variables. This leads to the following refinement of the information constraint (1.2) on a family of strategies $(\sigma_\eta)_\eta$. Note firstly that, given $c \in C_M$, with $X_1 = X \setminus \mathsf{Occ}_M(c)$ and $X_2 = \mathsf{Occ}_M(c)$, we can write $\eta \in \mathcal{I}^X$ as $\eta = (\eta_c, \eta_{\neg c})$, where $\eta_c \in \mathcal{I}^{X_1}$, $\eta_{\neg c} \in \mathcal{I}^{X_2}$. Now we can write the condition on $(\sigma_\eta)_\eta$ as follows:

$$\forall \eta, \eta' \in \mathcal{I}^X, c \in C_M, s \in \mathcal{D}(M). [\sigma_\eta(s)(c) = \sigma_{\eta_c, \eta'_{\neg c}}(\gamma_M(c)(s))(c)]. \quad (1.5)$$

(Here equality of partial functions is intended: either both states are undefined at c, or both are defined and have the same value.)

We now give the compositional definition of the occlusion function (non-trivial cases only).

1. **Choice connectives**.
$$\mathsf{Occ}_{M \oplus_\alpha N}(c) = \begin{cases} \varnothing, & c = c_0 \\ \mathsf{Occ}_M(c), & c \in C_M \\ \mathsf{Occ}_N(c), & c \in C_N. \end{cases}$$
2. **Quantifiers** $Q_\alpha x : X \to X \uplus \{x\}$.
$$\mathsf{Occ}_{Q_\alpha x}(c_0) = \varnothing.$$
3. **Parallel Composition** $\varphi \| \psi : X_1 \uplus X_2 \to Y_1 \uplus Y_2$, where $\varphi : X_1 \to Y_1$, $\psi : X_2 \to Y_2$. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$.
$$\mathsf{Occ}_{M \| N}(c) = \begin{cases} \mathsf{Occ}_M(c) \cup X_2, & c \in C_M \\ \mathsf{Occ}_N(c) \cup X_1, & c \in C_N. \end{cases}$$
4. **Sequential Composition** $\varphi \cdot \psi : X \to Z$, where $\varphi : X \to Y$ and $\psi : Y \to Z$. Let $M = [\![\varphi]\!]$, $N = [\![\psi]\!]$.
$$\mathsf{Occ}_{M \cdot N}(c) = \begin{cases} \mathsf{Occ}_M(c), & c \in C_M \\ (X \cap \mathsf{Occ}_N(c)) \cup (X \setminus Y), & c \in C_N. \end{cases}$$
5. **Role Interchange**. $\mathsf{Occ}_{\hat{\pi}(M)} = \mathsf{Occ}_M$.

The only case in the definition of the visibility function which needs to be revised to take account of the occlusion function is that for sequential composition:

$$\gamma_{M \cdot N}(c)(s) = \begin{cases} \gamma_M(c)(\pi_M(s)), & c \in C_M \\ (\pi_M(s) \setminus S) \cup \gamma_N(c)(\pi_N(s)), & c \in C_N \end{cases}$$

where

$$S = \{(c', v) \in D_M \mid \exists y \in (\mathsf{Occ}_N(c) \cap (Y \setminus X)). \, \mathsf{bind}_M(y) = c'\}.$$

IF-quantifiers. It is now a simple matter to extend the semantics to multi-agent versions of the IF-quantifiers. We consider a quantifier of the form $Q_\alpha x/Y : X \uplus Y \to X \uplus Y \uplus \{x\}$. Thus agent α is to make the choice for x, and must do so independently of what has been chosen for the variables in Y. The $\mathcal{A}$-game $M = [\![Q_\alpha x/Y]\!]$ is the same as for the standard quantifier $Q_\alpha x$, as are the bind and val functions. The difference is simply in the occlusion function:

$$\mathsf{Occ}_M(c_0) = Y.$$

This is then propagated by our compositional definitions into larger contexts in which the quantifier can be embedded, and feeds into the partial information constraint (1.5) to yield exactly the desired interpretation.

5 Compositionality Reconsidered

The issue of compositionality for IF-logic has attracted some attention[5]; see [Hi$_1$96]. In [Ho$_1$97a], Hodges gives a compositional Tarski-style semantics for IF logic. The key idea is to work at the level of sets of sets of assignments, rather than the sets of assignments used in the standard Tarskian semantics. It is not immediately clear how this relates to our compositional account.

In fact, this highlights an underlying issue which has perhaps thus far escaped the attention it deserves. One can distinguish two views on how Logic relates to Structure:

1. **The Descriptive View.** Logic is used to *talk about* structure. This is the view taken in Model Theory, and in most of the uses of Logic (Temporal logics, MSO etc.) in Verification in Computer Science. It is by far the more prevalent and widely-understood view.

2. **The Intrinsic View.** Logic is taken to *embody* structure. This is, implicitly or explicitly, the view taken in the Curry-Howard isomorphism, and more generally in Structural Proof Theory, and in (much of) Categorical Logic. In the Curry-Howard isomorphism, one is not using logic to *talk about* functional programming; rather, logic (in this aspect) *is* functional programming.

The present paper is largely informed by the second point of view, although we have also provided a basis for the first, in formulating a general compositional account of valuations, in terms of which validity and equilibrium notions can be formulated. Our purpose, as already remarked in the Introduction, is not to develop a modal logic (or similar) for talking *about* multi-agent games, but rather a logic whose semantics intrinsically *has* a structure of multi-agent games. This is in the same spirit as more familiar

[5] The discussion in this section was prompted by some insightful remarks and questions by one of the referees.

kinds of game semantics. Similarly, our account of IF constructs does not take the form of a logic which *talks about* agents with limited knowledge — hence some kind of epistemic logic — but rather a semantics which is intrinsically comprised of agents playing strategies subject to partial information constraints. These constraints were formalized in two distinct ways, the local and the global, in Section 3, and these two forms shown to be equivalent.

The compositionality in our account occurs at several levels, corresponding to the levels of a Curry-Howard style interpretation of a logic (although we have not formulated a proof system on the syntactic side here):

- Formulas are interpreted by games: this is the "static" part of our semantics, which is done compositionally.

- Proofs, or *witnesses* or *realizers* for the truth, or more generally the *outcomes* of formulas, are embodied as strategies for the various players. These strategies are defined compositionally in one of two ways: the *global* way, where the set of strategies for a given agent for each formula is defined in terms of the sets of strategies for the sub-formulas — this is analogous to realizability definitions; and the *local* way, where we take all strategies which satisfy the information constraints for the given agent — it is now these information constraints which must be defined compositionally.

- The dynamic aspect of this level of the semantics is given by formalizing the notion of playing the strategies in a profile off against each other to achieve an overall outcome. This definition, while made in a mathematically elegant denotational style, directly captures the operational content of the idea of agent interaction. It corresponds to the proof-theoretic dynamics of Cut-elimination, or to the realizability dynamics of applying a recursive function.

- Finally, the valuations of the outcomes, from the point of view of each agent, are defined compositionally. This requires a proper treatment of the binding of variables to cells, and hence the developments of Section 4.

6 Further Directions

There are numerous further directions which it seems interesting to pursue. We mention a few:

- Some extensions are quite straightforward. In particular, an extension of $\mathcal{L}_{\mathcal{A}}$ with *fixpoints*

$$\mu P(x_1, \ldots, x_n).\, \varphi(P)$$

can be considered. The standard theory of solutions of domain equations over CDS [Ka$_0$Pl$_1$93] can be used to extend the static semantics to such fixpoint formulas. Moreover, our semantic constructions work for arbitrary CDS with infinite states. The only point which needs to be reconsidered in this setting is how the valuation functions are defined. The best idea seems to be to define valuations on all states, not only maximal ones. The value space should itself include partial values and form a domain, and the valuation function should be continuous. For example, we could take $\mathcal{V}_\alpha$ to be the *interval domain* $\mathbb{I}[0,1]$ on the unit interval.

An extension to full second-order logic, although technically more demanding, is also possible [AbJa$_1$05].

- What is the full spectrum of possible connectives which can be used to explore the resources of our semantics? The logic $\mathcal{L}_\mathcal{A}$ we have introduced is quite natural, but this question remains wide open. Here is one precise version:

  Question 6.1. Which set of connectives and quantifiers is *descriptively complete*, in the sense that every finite CDS is the denotation of a formula built from these quantifiers and connectives?

  Another dimension concerns the information-flow structures and constraints expressible in the logic. The multiplicative connectives for sequential and parallel composition which we have studied are very basic. The parallel composition corresponds to the Linear Logic $\otimes$. The Linear Logic $⅋$ does allow for information flow between the parallel components; and there are surely a whole range of possibilities here.

  Problem 6.2. Classify the possibilities for multiplicative connectives and information-flow constraints in the semantic space of concurrent games and strategies.

  As one illustration, consider a connective $M \lhd N$ which combines features of sequential and parallel composition. The $\mathcal{A}$-game is defined as for $M \| N$, while the visibility function $\gamma_{M \lhd N}$ is defined as for $M \cdot N$. Play can proceed concurrently in both sub-games; there is information flow from M to N, but not *vice versa.*

- We have only considered *deterministic* strategies in this paper. Mixed, non-deterministic, probabilistic, and perhaps even quantum strategies should also be considered.

- The whole question of proof theory for the logic $\mathcal{L}_{\mathcal{A}}$ has been left open. In a sense, we have given a *semantics of proofs* without having given a syntax! How multi-agent proof theory should look is conceptually both challenging and intriguing.

- Viewed from a model-theoretic perspective, IF-logic seems dauntingly complex. Our more intensional and operational view may offer some useful alternative possibilities. Just as the shift from validity to model-checking often replaces an intractable problem by an efficiently solvable one, so the shift from model-theoretic validity or definability of IF-formulas to constructing, reasoning about and running strategies for concurrent games described by proofs of formulas seems a promising approach to making this rich paradigm computationally accessible.

- We may also seek to put our compositional analysis to use in analyzing game-theoretical equilibrium notions in a multi-agent, partial information setting. It may be that the tools we provide would facilitate an analysis by decomposition into subgames in a wider range of settings than classical game-theoretic methods allow.

- The logic and semantics we have developed appears to flow from very natural intuitions. These should be supported by a range of convincing examples and applications.

References

[Ab00a] S. Abramsky. Concurrent interaction games. In [$\mathrm{Da_0Ro_2Wo_2}$00, pp. 1–12].

[Ab00b] S. Abramsky. Process realizability. In [$\mathrm{Ba_8St_2}$00, pp. 167–180].

[Ab03] S. Abramsky. Sequentiality vs. concurrency in games and logic. *Mathematical Structures in Computer Science* 13(4):531–565, 2003.

[Ab06] S. Abramsky. Socially responsive, environmentally friendly logic. In [$\mathrm{Ah_1Pi_1}$06, pp. 17–45].

[$\mathrm{AbJa_1}$94] S. Abramsky & R. Jagadeesan. New foundations for the geometry of interaction. *Information and Computation* 111(1):53–119, 1994. Conference version appeared in LiCS 1992.

[$\text{AbJa}_1$05] S. Abramsky & R. Jagadeesan. A game semantics for generic polymorphism. *Annals of Pure and Applied Logic* 133(1–3):3–37, 2005.

[$\text{AbJa}_1\text{Ma}_1$00] S. Abramsky, R. Jagadeesan & P. Malacaria. Full abstraction for PCF. *Information and Computation* 163(2):409–470, 2000.

[$\text{AbMe}_1$99] S. Abramsky & P.-A. Melliès. Concurrent games and full completeness. In [$\text{Lo}_2$99, pp. 431–444].

[AbVi93] S. Abramsky & S. Vickers. Quantales, observational logic and process semantics. *Mathematical Structures in Computer Science* 3(2):161–227, 1993.

[$\text{Ah}_1\text{Pi}_1$06] T. Aho & A.-V. Pietarinen, eds. *Truth and Games: Essays in Honour of Gabriel Sandu*, *Acta Philosophica Fennica* 78. Societas Philosophica Fennica, 2006.

[Ap97] K.R. Apt. From chaotic iteration to constraint propagation. In [De_0Go_3MS97, pp. 36–55].

[$\text{ApPl}_1$81] K.R. Apt & G.D. Plotkin. A Cook's tour of countable nondeterminism. In [$\text{EvKa}_2$81, pp. 479–494].

[$\text{Ba}_4\text{Co}_0\text{Sa}_0$05] A. Baltag, B. Coecke & M. Sadrzadeh. Algebra and sequent calculus for epistemic actions. *Electronic Notes in Theoretical Computer Science* 126:27–52, 2005.

[$\text{Ba}_8\text{St}_2$00] F.L. Bauer & R. Steinbrüggen, eds. *Foundations of Secure Computation: Proceedings of the 1999 Marktoberdorf Summer School.* IOS Press, 2000.

[$\text{Bo}_3\text{So}_0$93] A.M. Borzyszkowski & S. Sokolowski, eds. *Mathematical Foundations of Computer Science 1993, 18th International Symposium, MFCS'93, Gdansk, Poland, August 30–September 3, 1993, Proceedings*, *Lecture Notes in Computer Science* 711. Springer, 1993.

[$\text{Da}_0\text{Ro}_2\text{Wo}_2$00] J. Davies, A.W. Roscoe & J. Woodcock, eds. *Millennial Perspectives in Computer Science.* Palgrave, 2000.

[De_0Go_3MS97] P. Degano, R. Gorrieri & A. Marchetti-Spaccamela, eds. *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7–11 July 1997, Proceedings*, *Lecture Notes in Computer Science* 1256. Springer, 1997.

[EvKa$_2$81] S. Even & O. Kariv, eds. *Automata, Languages and Programming, 8th Colloquium, Acre (Akko), Israel, July 13–17, 1981, Proceedings.* Springer, 1981.

[Fe$_2$Fr$_3$Hi$_0$89] J.E. Fenstad, I.T. Frolov & R. Hilpinen, eds. *Logic, Methodology and Philosophy of Science VIII*, *Studies in Logic and the Foundations of Mathematics* 126. Elsevier, 1989.

[Gi$_2$87] J.-Y. Girard. Linear logic. *Theoretical Computer Science* 50(1):1–101, 1987.

[He$_1$61] L. Henkin. Some remarks on infinitely long formulas. In *Infinitistic Methods: Proceedings of the Symposium on Foundations of Mathematics, Warsaw, 2–9 September 1959*, pp. 167–183. Pergamon Press and Państwowe Wydawnictwo Naukowe, 1961.

[Hi$_1$96] J. Hintikka. *The Principles of Mathematics Revisited.* Cambridge University Press, 1996.

[Hi$_1$Sa$_4$89] J. Hintikka & G. Sandu. Informational independence as a semantical phenomenon. In [Fe$_2$Fr$_3$Hi$_0$89, pp. 571–589].

[Hi$_1$Sa$_4$95] J. Hintikka & G. Sandu. What is the logic of parallel processing? *International Journal of Foundations of Computer Science* 6(1):27–49, 1995.

[Hi$_1$Sa$_4$96] J. Hintikka & G. Sandu. Game-theoretical semantics. In [vBtM96, pp. 361–410].

[Ho$_1$97a] W. Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL* 5(4):539–563, 1997.

[Hy$_0$On00] J.M.E. Hyland & C.-H.L. Ong. On full abstraction for PCF: I, II, and III. *Information and Computation* 163(2):285–408, 2000.

[Ja$_1$Pa$_2$Pi$_2$89] R. Jagadeesan, P. Panangaden & K. Pingali. A fully abstract semantics for a functional language with logic variables. In [Pa$_5$89, pp. 294–303].

[Ka$_0$Pl$_1$78] G. Kahn & G. Plotkin. Concrete domains. 336, IRIA-Laboria, 1978.

[$Ka_0Pl_1$93] G. Kahn & G. Plotkin. Concrete domains. *Theoretical Computer Science* 121(1–2):187–277, 1993.

[$Lo_2$99] G. Longo, ed. *14th Annual IEEE Symposium on Logic in Computer Science, 2–5 July, 1999, Trento, Italy.* IEEE, 1999.

[Mi93] R. Milner. Action calculi, or syntactic action structures. In [$Bo_3So_0$93, pp. 105–121].

[$MiPa_6Wa_1$92a] R. Milner, J. Parrow & D. Walker. A calculus of mobile processes, I. *Information and Computation* 100(1):1–40, 1992.

[$MiPa_6Wa_1$92b] R. Milner, J. Parrow & D. Walker. A calculus of mobile processes, II. *Information and Computation* 100(1):41–77, 1992.

[$Pa_5$89] R. Parikh, ed. *Proceedings, Fourth Annual Symposium on Logic in Computer Science, 5–8 June, 1989, Asilomar Conference Center, Pacific Grove, California, USA*. IEEE Computer Society, 1989.

[$Pa_7$02] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation* 12(1):149–166, 2002.

[$Pa_7Pa_5$03] M. Pauly & R. Parikh. Game logic — an overview. *Studia Logica* 75(2):165–182, 2003.

[$Sa_5Ri_0Pa_2$91] V.A. Saraswat, M.C. Rinard & P. Panangaden. Semantic foundations of concurrent constraint programming. In [Wi_3+91, pp. 333–352].

[vB03] J. van Benthem. Logic games are complete for game logics. *Studia Logica* 75(2):183–203, 2003.

[vBtM96] J. van Benthem & A. ter Meulen, eds. *Handbook of Logic and Language.* Elsevier, 1996.

[Wi_3+91] D. Wise, C. Mann, J. Graver & R. Cartwright, eds. *Conference Record of the Eighteenth Annual ACM Symposium on Principles of Programming Languages.* ACM Press, January 1991.

Quantificational Modal Operators and Their Semantics[*]

Stefano Borgo

Laboratory for Applied Ontology
Institute for Cognitive Sciences and Technology
Via alla Cascata 56/C
38100 Povo (Trento), Italy
borgo@loa-cnr.it

Abstract

We study a system of modal logic for representing and reasoning in multi-agent systems. Building on dynamic action logic and Henkin quantifiers, we introduce a class of operators that present important features for capturing concurrency, independence, collaboration, and co-ordination between agents. The main goal of the paper is to introduce the formal semantics of these operators and to show how they can model different types of agents. This yields a way to directly compare a variety of phenomena in multi-agent systems. Some examples are given.

1 Introduction

For about 20 years we have witnessed an increasing interest in the formal study of phenomena comprising several entities which present independent and autonomous behavior like software agents, human beings, biological entities, social organizations, robots and stock markets [We$_1$99].

In this research area, the issue of (true) concurrency has special interest since it puts at the center phenomena where several entities act simultaneously perhaps affecting each other. This issue is coupled with the need to formalize group collaboration as well as group dynamics. Another challenge is the formalization of the independence of an agent from the others and from the groups of which it is a member. Modelling the choices an agent makes or can make depending on its "internal" status, its beliefs about the external world, its knowledge about (and relationship with) other entities/agents is at the center of many representation problems.

The usual logical machinery often requires the coexistence of logical operators (dynamic, temporal, epistemic, and deontic) in one and the same

[*] The author has been supported by the projects TOCAI.IT (MIUR), MOSTRO and LIFOA (Provincia Autonoma di Trento).

language [vHVe$_2$02, vHWo$_3$03]. This strategy is not satisfactory because of the complexity of the logical systems thus obtained [Be$_0$+02, vHWo$_3$03]. Furthermore, these logics are hard to compare to the point that the uniformity of the very phenomena at stake is lost in the different formalizations. An example is given by logics like **ATL** [Al$_2$He$_2$Ku$_1$02], **CL**, **ECL** [Pa$_7$02], **ATEL** [vHWo$_3$02], and **STIT** logic [Ho$_6$01, Br$_2$He$_3$Tr05] which deal with roughly the same type of scenario [Go$_2$Ja$_2$04, Wö04].

Our work focuses on the formalization of multi-agent systems with particular emphasis on concurrency, independence, collaboration, and coordination issues. Our ideal scenario comprises a fixed set of agents that individually or in group, isolately or co-ordinated, take actions simultaneously and in this way determine the transitions between states of the system. The main goal of the paper is to show that the language we have developed has several natural interpretations which allow us to capture different types of agents while maintaining the very same syntax. The novelty is given by a new type of operators that combines modality with quantification. For this reason, these operators are called *quantificational modal operators*.

In this approach, we take a general perspective and do not limit our work to a specific notion of agent (and so we are not going to give one). Note that, for presentation purposes, we often describe the agents as having some degree of rationality. This is not necessary but it helps in conveying the meaning of the operators. Also, in this paper we do not discuss proof-theoretical properties of the resulting logics. These interpretations correspond to quite different axiomatic systems and here we lack the space for their analysis. The interested reader can find in [Bo$_1$05b] an axiomatic presentation of one of these systems. Finally, note that in this study we shall always stick to two-valued semantics.

Structure of the paper. Section 2 first introduces the propositional fragment of the logic by defining the constant modal operators and, secondly, extends them with free variables. In Section 3, we present the full language by introducing the quantificational modal operators and then study alternative interpretations for one-column operators. In Section 4 we briefly discuss the extension of these semantics to multi-column operators. The following section looks at a couple of examples. Finally, Section 6 relates our work to other logical approaches and adds some final remark.

2 Basic modalities for MAS

The modalities we want to study can be seen as an extension of *Dynamic Action Logic*, that is, the application of Dynamic Logic (**DL**) [Ha$_1$Ko$_6$Ti$_1$00] to model actions. Similarly to **DL**, our basic operators, called *constant modal operators*, are modalities indexed by constant identifiers (denoting actions). However, these operators differ from those of **DL** in two aspects:

syntactically they require several constant identifiers to individuate even the simplest modalities, and semantically they are not associated to a unique interpretation.

In a system of two agents, say $\mathcal{A}_1$ and $\mathcal{A}_2$, our modal operators have the shape of a $2 \times n$ matrix ($n > 0$) where the first row lists the (constants denoting the) actions performed by agent $\mathcal{A}_1$, in the order of their execution, and the second row lists the actions performed by agent $\mathcal{A}_2$. For instance the expression $\left[\begin{smallmatrix} c_1 & c_3 \\ c_2 & c_4 \end{smallmatrix}\right]$ is a modality corresponding to the state transition identified by the concurrent execution of action c_1 (by agent $\mathcal{A}_1$) and c_2 (by agent $\mathcal{A}_2$) followed by the concurrent execution of action c_3 (by agent $\mathcal{A}_1$) and c_4 (by agent $\mathcal{A}_2$). Each entry of the matrix denotes an action and the combination of these actions characterises the meaning of the modal operator by identifying, in the usual Kripke style semantics, the accessibility relation associated with that operator.

More generally, an operator in the shape of a $k \times n$ matrix is a modality for a system with k agents. It is always assumed that the number of rows in the operators matches the number of agents in the system (as a consequence all the operators in a language have the same number of rows). Also, each agent is associated to the same row in all operators.

We now state this formally:

Let PropId be a non-empty countable set, the set of *proposition identifiers*. Let ActId (disjoint from PropId) be a non-empty countable set whose elements are called *action identifiers*. These are the individual constants of the language. Following standard modal logic, complex formulas are generated inductively from proposition identifiers through the connectives of implication ($\rightarrow$) and negation ($\neg$), and the modal operators described below. As usual, we shall make use of the standard conventions for $\wedge, \vee, \leftrightarrow$.

Fix an integer $k \geq 1$ which, informally, is the number of agents in the system. A *constant modality marker*[1] for k is a $k \times n$-matrix ($n \geq 1$)

$$M = \begin{matrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kn} \end{matrix}$$

where $a_{ij} \in$ ActId (a_{ij}, a_{mn} not necessarily distinct).

A *constant modal operator* for k is an expression $[M]$ where M is a constant modality marker for k.

The set of k-formulas (formulas for short) is the smallest set F_k satisfying:

I) PropId $\subseteq F_k$ (the elements of PropId are called atomic formulas),

II) If φ and ψ are in F_k, then so are $\neg\varphi$ and $\varphi \rightarrow \psi$,

[1] Elsewhere we have been calling these *constant modality identifiers.*

III) If $[M]$ is a constant modal operator for k and φ is in F_k, then $[M]\varphi$ is in F_k also.

The semantics is as follows:

Fix a set Act of actions, we call *k-action* an expression in the shape of a $k \times n$ matrix ($n \geq 1$) over Act. A *k-agent Kripke frame* is a triple $\mathcal{K} = \langle W, \mathrm{Act}; R\rangle$ where W is a non-empty set *(the set of states)*, Act is a non-empty set *(the set of actions)*, and R is a function mapping k-actions (over Act) of size $k \times 1$ to binary relations on W, $R\begin{pmatrix}\alpha_1\\ \vdots\\ \alpha_k\end{pmatrix} \subseteq W \times W$.

A *k-agent Kripke structure* is a tuple $\mathcal{M} = \langle W, \mathrm{Act}; R, [\![\cdot]\!]\rangle$ where $\langle W, \mathrm{Act}; R\rangle$ is a k-agent Kripke frame and $[\![\cdot]\!]$ is a function *(the valuation function)* such that $[\![p]\!] \subseteq W$ for $p \in \mathrm{PropId}$ and $[\![a]\!] \in \mathrm{Act}$ for $a \in \mathrm{ActId}$.

Let us write $\mathcal{A}_1$ for the first agent, $\ldots, \mathcal{A}_k$ for the kth agent. If agent $\mathcal{A}_1$ performs the action denoted by a_1, agent $\mathcal{A}_2$ the action denoted by $a_2, \ldots,$ agent $\mathcal{A}_k$ the action denoted by a_k, we write $\begin{bmatrix}a_1\\ a_2\\ \vdots\\ a_k\end{bmatrix}$ for the modal operator describing the evolution of the system which is determined by the concurrent execution of actions $[\![a_1]\!], \ldots, [\![a_k]\!]$ by agents $\mathcal{A}_1, \ldots, \mathcal{A}_k$, respectively. That is, the interpretation of $\begin{bmatrix}a_1\\ a_2\\ \vdots\\ a_k\end{bmatrix}$ is k-action $\begin{matrix}[\![a_1]\!]\\ [\![a_2]\!]\\ \vdots\\ [\![a_k]\!]\end{matrix}$.

Function $[\![\cdot]\!]$ is extended inductively to multi-column operators in the language as follows: if $[A]$ is a multi-column operator obtained by juxtaposition of constant modality markers B and C (i.e. $[A] = [BC]$), then we put $R([\![A]\!]) = R([\![B]\!]) \circ R([\![C]\!])$. More formally,

$$\left[\!\!\left[\begin{matrix}a_{11}\\ a_{21}\\ \vdots\\ a_{k1}\end{matrix}\right]\!\!\right] =_{\mathrm{def}} \begin{matrix}[\![a_1]\!]\\ [\![a_2]\!]\\ \vdots\\ [\![a_k]\!]\end{matrix} \quad ; \quad \left[\!\!\left[\begin{matrix}a_{11} & a_{12} & \cdots & a_{1n}\\ a_{21} & a_{22} & \cdots & a_{2n}\\ \vdots & \vdots & & \vdots\\ a_{k1} & a_{k2} & \cdots & a_{kn}\end{matrix}\right]\!\!\right] =_{\mathrm{def}} \left[\!\!\left[\begin{matrix}a_{11}\\ a_{21}\\ \vdots\\ a_{k1}\end{matrix}\right]\!\!\right] \left[\!\!\left[\begin{matrix}a_{12}\\ a_{22}\\ \vdots\\ a_{k2}\end{matrix}\right]\!\!\right] \cdots \left[\!\!\left[\begin{matrix}a_{1n}\\ a_{2n}\\ \vdots\\ a_{kn}\end{matrix}\right]\!\!\right]$$

Note that we write $[\![M]\!]$ instead of $[\![[M]]\!]$.

The truth value of a formula is defined inductively:

1. Let $p \in \mathrm{PropId}$, then $\mathcal{M}, s \models p$ if $s \in [\![p]\!]$
2. $\mathcal{M}, s \models \neg\varphi$ if $\mathcal{M}, s \not\models \varphi$
3. $\mathcal{M}, s \models \varphi \to \psi$ if $\mathcal{M}, s \not\models \varphi$ or $\mathcal{M}, s \models \psi$
4. $\mathcal{M}, s \models [A]\varphi$ if $\mathcal{M}, t \models \varphi$ for all $t \in W$ such that $(s,t) \in R([\![A]\!])$

A *k-agent Kripke model* for a set of formulas Σ in the language is a k-agent Kripke structure $\mathcal{M}$ such that all formulas $\varphi \in \Sigma$ hold in all states of $\mathcal{M}$ (i.e., are *valid* in $\mathcal{M}$).

As anticipated, the language here presented modifies the basic fragment of **DL**. However, the major novelty, we believe, lies in the change of perspective it pushes for: we need k action identifiers (or multiples of k) to describe the evolution of the whole system since only the combination of all *concurrent actions* can provide this information. Also, this formalism provides a more acceptable notion of action since the outcome of the execution of $\alpha \in \mathrm{Act}$ by an agent is not determined by α alone.

Now we extend the constant operators by allowing the occurrence of free variables. This extension is a first step to introduce quantificational operators, a move we shall motivate in next section.

Fix a new set $\mathrm{Var} = \{x, y, z, \ldots\}$ of variables. Let $\Im$ be an environment function from the set Var to the set Act and let a *modality marker* be any $k \times n$ matrix defined as before but this time with condition $a_{i,j} \in \mathrm{ActId} \cup \mathrm{Var}$ (for all relevant indices i, j). The extension of the set of k-formulas F_k to include these modalities is trivial. Their interpretation requires the new function $\Im$, that is, now relation $\models$ is defined over the triple $\mathcal{M}, s, \Im$. For instance, clause 4. becomes: $\mathcal{M}, s, \Im \models [M]\varphi$ if $\mathcal{M}, t, \Im \models \varphi$ for all $t \in W$ such that $(s,t) \in R([\![M]\!])$ where $[\![x]\!] = \Im(x)$ when $x \in \mathrm{Var}$. The remaining clauses are analogous to 1–3 above.

3 Quantificational modal operators

Our next goal is the introduction of quantificational modalities in the logic of Section 2. The basic idea is to enrich modality markers with a form of generalized quantifiers introduced by Henkin in [He$_1$61].

Henkin quantifiers are matrices of standard quantifier prefixes[2] such as

$$\begin{pmatrix} \forall x_1 & \exists x_2 & \exists x_3 \\ \exists y_1 & \forall y_2 & \exists y_3 \end{pmatrix}. \tag{1.1}$$

Syntactically these are unary operators, that is, if (H) is a Henkin quantifier and φ is a formula, then $(H)\varphi$ is a formula as well. There is no restriction on the number or position of the quantifiers $\forall$ and $\exists$ in the matrix but no variable may occur more than once.

It is well known that Henkin quantifiers are more expressive than standard quantifiers [Kr$_2$Mo$_4$95] and we take advantage of their strength to ensure row independence in the modalities (which, in turn, models the independence of the agents from each other). Consider a modality with

[2] We follow common practice and use the term 'Henkin quantifiers' although these are, properly said, Henkin prefixes. Also, note that the matrix form can be relaxed as we do in formula (1.3) below.

constants and free variables (as described at the end of Section 2), say

$$\begin{bmatrix} x_1 & a & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} p_0. \tag{1.2}$$

The occurrence of a free variable in entry $A(i, j)$ of the matrix suggests that the jth action that agent $\mathcal{A}_i$ is going to perform has not been fixed. One can leave it undetermined meaning that this action depends on the model's environment function. Alternatively, one may want the agent herself to decide which action to perform. In the latter situation, we want to model whether the agent decides in favor or against the realization of p_0 since p_0 is implicitly 'proposed as a goal' by that modal formula. For this reason, we combine modal operators and (a version of) Henkin quantifiers as in the following expression

$$\begin{pmatrix} & & \exists x_3 \\ \exists y_1 & \forall y_2 & \exists y_3 \end{pmatrix} \begin{bmatrix} x_1 & a & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} p_0 \tag{1.3}$$

where, of course, each agent is associated to the same row in both the Henkin quantifier and the modality. In this expression, a free variable x_h in position (i, j) indicates that the agent $\mathcal{A}_i$ at step j performs the action $\Im(x_h)$, i.e., the action determined by the model's environment function. A constant c in position (i, j) indicates that the agent $\mathcal{A}_i$ at step j performs the action $[\![c]\!]$. Finally, a quantified variable x_h in position (i, j) indicates that the agent $\mathcal{A}_i$ chooses what to perform at step j and the specific quantifier marks the attitude of that agent (at this time-step) toward formula p_0. More precisely, if $\exists x_h$ occurs, at time-step j agent $\mathcal{A}_i$ chooses an action with the intention of making p_0 true. Instead, if $\forall x_h$ occurs, the same agent chooses an action randomly.[3]

Here we propose a restriction of this language that consists in merging Henkin quantifiers and modality markers into a unique operator, called *quantificational modal operator*. Thus, instead of formula (1.3) we write

$$\begin{bmatrix} x_1 & a & \exists x_3 \\ \exists y_1 & \forall y_2 & \exists y_3 \end{bmatrix} p_0. \tag{1.4}$$

Following the above discussion, we shall say that p_0 is a goal for agent $\mathcal{A}_i$ at time j if an existentially quantified variable occurs at position (i, j) of the modality, and that p_0 is not a goal (at that time for that agent) if an universally quantified variable occurs instead.

Definition 3.1. A *quantificational modality marker* is a $k \times n$ matrix with each entry containing an action identifier, a variable, or a quantified variable

[3] Informally, she chooses according to her goals. The occurrence of '$\forall$' tells us that p_0 is not a goal for this agent when choosing at this position in the matrix. Thus, from the local perspective given by the formula one can think that the agent chooses randomly at time-step j. A view that further justifies our adoption of the symbols $\forall$ and $\exists$.

provided a variable occurs at most once in a marker. A *quantificational* (*modal*) *operator* is an expression $[M]$ where M is a quantificational modality marker.

We write QOP for the set of quantificational modal operators, OP for the quantificational operators where no variable occurs quantified.

The set F_k of k-formulas is defined as in Section 2 but in clause III) we now use the larger class of quantificational operators. The scope of the modal operator is the formula to which it applies. The scope of a quantifier in a modal operator is the scope of the modal operator itself.[4] Note that we inherit from the Henkin quantifiers the general proviso on variable occurrence in quantificational modality markers (and equivalently in quantificational modal operators).

It remains to discuss the semantics of this language. Below, we investigate alternative interpretations for the quantificational operators starting with one column modalities. We anticipate, at the informal level, that the interpretation in a structure $\mathcal{M} = \langle W, \mathrm{Act}; R, [\![\cdot]\!]\rangle$ of a quantificational operator, say $\begin{bmatrix} x_1 & a & \exists x_3 \\ \exists y_1 & \forall y_2 & \exists y_3 \end{bmatrix}$, takes two steps. In the first step one interprets formula $\begin{pmatrix} & & \exists x_3 \\ \exists y_1 & \forall y_2 & \exists y_3 \end{pmatrix} \varphi$ with $\varphi = \begin{bmatrix} x_1 & a & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} p_0$. The second step amounts to the evaluation of the formula $\begin{bmatrix} x_1 & a & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} p_0$ obtained by using the values chosen at the first step for the interpretation of the bound variables. The precise formulation in different cases (all restricted to one-column operators) is given below. Note also that we restrict our examples to two-agent systems. However, the argument is easily generalized to an arbitrary number of agents.

3.1 Risk-averse co-ordinated agents

We begin with an interpretation that follows from the classical meaning of the quantifiers $\exists$ and $\forall$. This view relies on what actions exist without considering agents' strategies or capacities.

Fix a structure $\mathcal{M}$ for two agents, say $\mathcal{A}_1$ and $\mathcal{A}_2$, and let $[\![b]\!] = \beta$. First, we look at $\begin{bmatrix} \exists x \\ b \end{bmatrix} p_0$. This formula holds at a state s if and only if there exists an action α such that p_0 is true at all states t with $(s,t) \in R\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$. Such a formula is read: "there exists an action α such that after agent $\mathcal{A}_1$ executes α and (concurrently) agent $\mathcal{A}_2$ executes β, p_0 holds". Similarly, formula $\begin{bmatrix} \forall x \\ b \end{bmatrix} p_0$ corresponds to: "for any action α, after agent $\mathcal{A}_1$ executes it and

[4] It follows from the previous discussion that a quantified variable in the modal operator stands for a quantifier prefix (bounding the occurrences of the variable in the scope of the modality) and for a bound occurrence of that very variable (whose value is needed in this position to interpret the modal operator itself).

(concurrently) agent $\mathscr{A}_2$ executes β, p_0 holds" since it is true at a state s if and only if for all actions α, p_0 is true at all states t with $(s,t) \in R\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$.

If we assume that the agents form a coalition or, more generally, are coordinated whenever they both have p_0 as goal, we have that $\begin{bmatrix} \exists x \\ \exists y \end{bmatrix} p_0$ is true if there exist actions α and β (not necessarily distinct) such that p_0 holds in all states reachable through $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$. The meaning of formula $\begin{bmatrix} \forall x \\ \forall y \end{bmatrix} p_0$ is now obvious: it is true if p_0 is true in *any* state reachable from s via *any* transition.

An important issue arises when considering operators where both quantifiers occur as, for instance, in formula $\begin{bmatrix} \forall x \\ \exists y \end{bmatrix} p_0$. To establish the truth value of this formula at a given state we have two options. One can verify that a value β for y exists such that p_0 is true in all states reachable through $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ for any α. An alternative is to state the formula true if for every action α, there exists β such that p_0 is true in all states t with $(s,t) \in R\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$.

By embracing the first interpretation, one extends the semantics of Section 2 with the following clause for quantificational *one-column* operators (for multi-column operators further issues must be addressed, see Section 4):

5_1. Let $[X]$ be a quantificational operator with existentially quantified variables $x_1, \ldots, x_r$ and universally quantified variables $y_1, \ldots, y_s$ $(r, s \geq 0)$. Then $\mathcal{M}, s, \Im \models [X]\varphi$ if: there exist $\alpha_1, \ldots, \alpha_r \in$ Act such that for all $\beta_1, \ldots, \beta_s \in$ Act, if Γ is the k-action obtained by substituting α_i for $\exists x_i$, β_j for $\forall y_j$, and $[\![c_h]\!]$ for each action identifier or free variable c_h in $[X]$ (for all relevant indices i, j, h), then for all $(s,t) \in R(\Gamma)$, $\mathcal{M}, t, \Im^* \models \varphi$ where $\Im^*(x_i) = \alpha_i$, $\Im^*(y_j) = \beta_j$, and $\Im^*(z) = \Im(z)$.

This semantic clause puts strong constrains on the (one-column) modal operators to the point that it suffices to enrich the basic language of Section 2 with the quantifiers of standard first-order logic to eliminate the need of quantificational modalities. Indeed, clause 5_1 corresponds to the interpretation obtained by replacing quantified variables in the operator with a sequence of standard quantifiers as shown, in a two-agent system, by the following function τ_1:

$$p \overset{\tau_1}{\longmapsto} p \quad \text{(for } p \text{ atomic)} \tag{1_1}$$

$$\neg\varphi \overset{\tau_1}{\longmapsto} \neg\tau_1(\varphi) \tag{2_1}$$

$$\varphi \to \psi \overset{\tau_1}{\longmapsto} \tau_1(\varphi) \to \tau_1(\psi) \tag{3_1}$$

$$\begin{bmatrix} \forall x \\ \forall y \end{bmatrix} \varphi \overset{\tau_1}{\longmapsto} \forall x, y \begin{bmatrix} x \\ y \end{bmatrix} \tau_1(\varphi) \qquad (4_1)$$

$$\begin{bmatrix} \exists x \\ \forall y \end{bmatrix} \varphi \overset{\tau_1}{\longmapsto} \exists x \forall y \begin{bmatrix} x \\ y \end{bmatrix} \tau_1(\varphi) \qquad (5_1)$$

$$\begin{bmatrix} \forall x \\ \exists y \end{bmatrix} \varphi \overset{\tau_1}{\longmapsto} \exists y \forall x \begin{bmatrix} x \\ y \end{bmatrix} \tau_1(\varphi) \qquad (6_1)$$

$$\begin{bmatrix} \exists x \\ \exists y \end{bmatrix} \varphi \overset{\tau_1}{\longmapsto} \exists x, y \begin{bmatrix} x \\ y \end{bmatrix} \tau_1(\varphi) \qquad (7_1)$$

We dub the agents satisfying clause 5_1 the *risk-averse co-ordinated agents*: "risk-averse" because they choose independently of others' decisions as τ_1 makes clear in (5_1) and (6_1). They are "co-ordinated" because, whenever they have a common goal, if possible they execute actions that combined allow them to reach that goal: case (7_1). Indeed, if agents $\mathscr{A}_1$ and $\mathscr{A}_2$ aim at making a formula true, then they behave like a coalition.

Note that the following formula-schema holds for clause 5_1:

$$\begin{bmatrix} \vdots \\ \forall x \\ \vdots \end{bmatrix} p_0 \to \begin{bmatrix} \vdots \\ \exists x \\ \vdots \end{bmatrix} p_0.$$

3.2 Isolated agents

Let us go back to formula $\begin{bmatrix} \exists x \\ b \end{bmatrix} p_0$. We now want to interpret this formula as saying that at a state s agent $\mathscr{A}_1$ *can* choose an action α such that p_0 is true at t for all $(s,t) \in R\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$. That is, this time formula $\begin{bmatrix} \exists x \\ b \end{bmatrix} p_0$ corresponds to reading "agent $\mathscr{A}_1$ can choose an action such that after agent $\mathscr{A}_1$ executes it and (concurrently) agent $\mathscr{A}_2$ executes β, p_0 holds".

Regarding formula $\begin{bmatrix} \forall x \\ b \end{bmatrix} p_0$, informally here we read it as follows: "no matter the action that agent $\mathscr{A}_1$ can choose, after agent $\mathscr{A}_1$ executes it and (concurrently) agent $\mathscr{A}_2$ executes β, p_0 holds". Since we do not put restrictions on the actions an agent can choose or execute, all the actions are possible and must be considered to evaluate the truth value of this formula, i.e., we end up with the following reading: "after agent $\mathscr{A}_1$ executes some action and (concurrently) agent $\mathscr{A}_2$ executes β, p_0 holds".

The meaning of $\begin{bmatrix} \forall x \\ \forall y \end{bmatrix} p_0$ is quite natural at this point: "no matter which action agent $\mathscr{A}_1$ executes and (concurrently) which action agent $\mathscr{A}_2$ executes, p_0 holds in the reached states". That is, with the above assumptions, the notion of 'choice' does not affect the meaning of '$\forall$'. For operators where both quantifiers occur, consider first $\begin{bmatrix} \forall x \\ \exists y \end{bmatrix} p_0$. Here if the agents choose independently (not knowing each other's doing), for the formula to be true the second agent has to find an action β such that for all actions α and all $(s,t) \in R\binom{\alpha}{\beta}$, p_0 holds at t. Analogously, for $\begin{bmatrix} \exists x \\ \forall y \end{bmatrix} p_0$.

Finally, formula $\begin{bmatrix} \exists x \\ \exists y \end{bmatrix} p_0$ is true if the agents can choose actions, say α and β (possibly the same), such that p_0 is true at any state t such that $(s,t) \in R\binom{\alpha}{\beta}$, i.e. "for all choices α made by $\mathcal{A}_1$ and all choices β made by $\mathcal{A}_2$, after $\mathcal{A}_1$ has executed α and $\mathcal{A}_2$ has (concurrently) executed β, p_0 holds". From our reading, we know that $\mathcal{A}_1$ and $\mathcal{A}_2$ have p_0 as (common) goal and, similarly to Section 3.1, we may further establish that they co-operate (or not). However, now we are not in a position to provide a formal definition yet. It remains to be explained what it means that the very agents 'can choose'.

For the time being, let us assume that

1) all elements in the quantificational modal operators (in particular, all action identifiers) are known to all agents

2) all agents choose independently and without communicating (in particular, they do not co-operate nor co-ordinate)

The goal is to extend the semantics of Section 2 to formulas with quantificational operators in such a way that these assumptions are captured.

First, we fix a new function $\mathfrak{C}$, called *choice function*. The intent is that $\mathfrak{C}$ codifies the behavior of the agents by providing the choices the agents make. Function $\mathfrak{C}$ takes as arguments: (1) the modal operator, (2) its scope formula and (3) the variable x, which implicitly gives the agent's index i.[5] Also, since the choices of the agents may depend on their knowledge about the state of the system and other agents' actions, we furnish $\mathfrak{C}$ with two further arguments: (4) the actual state w and (5) subsets of $\{1,\ldots,k\} \times \mathrm{Var} \times \mathrm{Act}$. The reason for this last argument will be discussed in Section 3.3. On input $([M], \varphi, x, w, K)$, with $K \subset \{1,\ldots,k\} \times \mathrm{Var} \times \mathrm{Act}$, $\mathfrak{C}$ returns pairs $(x,\alpha) \in \mathrm{Var} \times \mathrm{Act}$.[6] Thus, given a variable x in row i of a formula φ, $\mathfrak{C}$ provides the agent $\mathcal{A}_i$'s choice(s) for this variable taking into account other information like the actual state. (Since argument (5) is not relevant in this section, for the time being we take $K = \varnothing$.)

[5] More generally (as will be seen later), $\mathfrak{C}$ takes sets of variables as third argument thus we should write $\{x\}$.

[6] For each x occurring in the third argument, $\mathfrak{C}$ returns one or more pairs (x,α) with $\alpha \in \mathrm{Act}$. Admittedly, one may want to generalize the choice function even further to capture some special case. However, the one we have introduced here suffices for a large class of multi-agent systems.

5_2. Let $[X]$ be a quantificational operator with existentially quantified variables $x_1, \dots, x_r$ and with universally quantified variables $y_1, \dots, y_s$ $(r, s \geq 0)$. Then, $\mathcal{M}, s, \Im \models [X]\varphi$ if: for all $\alpha_1, \dots, \alpha_r \in$ Act such that $\alpha_i \in \mathfrak{C}([X], \varphi, x_i, s, \varnothing)$ and for all $\beta_1, \dots, \beta_s \in$ Act, if Γ is the k-action obtained by substituting α_i for $\exists x_i$, β_j for $\forall y_j$, and $[\![c_h]\!]$ for each action identifier or free variable c_h in $[X]$ (for all relevant indices i, j, h), then for all $(s,t) \in R(\Gamma)$, $\mathcal{M}, t, \Im^* \models \varphi$ where $\Im^*(x_i) = \alpha_i$, $\Im^*(y_j) = \beta_j$, and $\Im^*(c_h) = \Im(c_h)$.

Let $\vec{\alpha} = \alpha_1, \dots, \alpha_r$ and $\vec{\alpha}' = \alpha'_1, \dots, \alpha'_r$ be two r-tuples in Act. A *fusion* of $\vec{\alpha}$ and $\vec{\alpha}'$ is a r-tuple $\vec{\alpha}'' = \alpha''_1, \dots, \alpha''_r$, where $\alpha''_i \in \{\alpha_i, \alpha'_i\}$.

Proposition 3.2. If both $\vec{\alpha}$ and $\vec{\alpha}'$ satisfy the condition in clause 5_2, then any fusion of $\vec{\alpha}, \vec{\alpha}'$ satisfies it as well.

Informally, this property shows that the agents described by clause 5_2 cannot communicate and, consequently, we say that the agents described by this clause are *isolated.*

Unlike in the previous section, in the semantics given by 5_2 formulas

$$\begin{bmatrix} \vdots \\ \exists x \\ \vdots \\ \forall y \\ \vdots \end{bmatrix} p_0 \rightarrow \begin{bmatrix} \vdots \\ \exists x \\ \vdots \\ \exists y \\ \vdots \end{bmatrix} p_0 \ ; \quad \begin{bmatrix} \vdots \\ \exists x \\ \vdots \\ a \\ \vdots \end{bmatrix} p_0 \rightarrow \begin{bmatrix} \vdots \\ \exists x \\ \vdots \\ \exists y \\ \vdots \end{bmatrix} p_0 \tag{1.5}$$

are not valid since the choice function $\mathfrak{C}$ may be sensitive to the occurrences of quantifiers in $[X]$. This result is motivated by the following scenarios.

Two people, $\mathcal{R}_1$ and $\mathcal{R}_2$, are celebrating some achievement. $\mathcal{R}_1$ brought a cake for the occasion. We write p_0 for "the cake is sliced."

In the first scenario, $\mathcal{R}_2$ does not care about cutting the cake and this is known to $\mathcal{R}_1$. A formula that correctly models this situation contains a quantificational operator with an existential quantifier in the first row (the row associated to $\mathcal{R}_1$). The different attitude of $\mathcal{R}_2$ is described by the occurrence of an universal quantifier in the second row. The formula is: $\begin{bmatrix} \exists x \\ \forall y \end{bmatrix} p_0$. Things change if $\mathcal{R}_2$ also wants the cake to be cut. This second scenario is described by the formula $\begin{bmatrix} \exists x \\ \exists y \end{bmatrix} p_0$.

We now use these scenarios to prove that formula $\begin{bmatrix} \exists x \\ \forall y \end{bmatrix} p_0 \rightarrow \begin{bmatrix} \exists x \\ \exists y \end{bmatrix} p_0$ fails. In the antecedent $\begin{bmatrix} \exists x \\ \forall y \end{bmatrix} p_0$, $\mathcal{R}_1$ has the goal of getting the cake cut. Since the other agent does not have such a goal, $\mathcal{R}_1$ chooses to execute

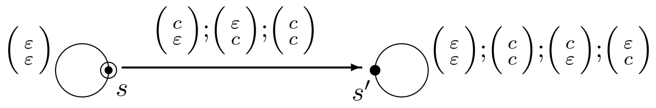

$\binom{\varepsilon}{\varepsilon}$
$\binom{c}{\varepsilon};\binom{\varepsilon}{c};\binom{c}{c}$
$\binom{\varepsilon}{\varepsilon};\binom{c}{c};\binom{c}{\varepsilon};\binom{\varepsilon}{c}$
s
s'

FIGURE 1. The 'cake slicing' frame (The actual state is double circled.)

the action 'cut the cake' which ensures the satisfaction of p_0 in the next state. In the consequent, $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same goal and this is common knowledge because of point 1) of page 58. Since they both have the same goal, $\mathcal{A}_1$ chooses not to cut the cake to let $\mathcal{A}_2$ do the honors. For the same reason, $\mathcal{A}_2$ decides not to cut the cake. Since nobody performs the action of cutting the cake (recall this is a single time-step with concurrent actions) the consequent formula turns out to be false. (Clearly, one can reformulate the example using an action like 'write in memory slot #321' which fails whenever two agents try to perform it at the same time.)

Let us see how function $\mathfrak{C}$ looks for these agents.

The attitude of both agents can be described by the informal rule "cut the cake unless somebody else is willing to do it". In a structure as depicted in Figure 1 where p_0 is false at s and true at s', and the possible actions are c, ε (c stands for "cut the cake" and ε for "do nothing"), function $\mathfrak{C}$ is given by

- $\mathfrak{C}\left(\begin{bmatrix}\exists x\\ \forall y\end{bmatrix}, p_0, x, s, \varnothing\right) = \{(x, c)\}$
  (agent $\mathcal{A}_1$, who has to decide the value of x, chooses c since agent $\mathcal{A}_2$ is not committed to get the cake cut as recognizable by the universal quantifier in the second row);

- $\mathfrak{C}\left(\begin{bmatrix}\exists x\\ \forall y\end{bmatrix}, p_0, y, s, \varnothing\right) = \{(y, c), (y, \varepsilon)\}$
  (agent $\mathcal{A}_2$ may perform any action since p_0 is not her goal);

- $\mathfrak{C}\left(\begin{bmatrix}\exists x\\ \exists y\end{bmatrix}, p_0, x, s, \varnothing\right) = \{(x, \varepsilon)\}$
  (agent $\mathcal{A}_1$ decides not to cut the cake: $\mathcal{A}_2$ is going to ensure it since it is her goal as recognizable by the existential quantifier in the second row);

- $\mathfrak{C}\left(\begin{bmatrix}\exists x\\ \exists y\end{bmatrix}, p_0, y, s, \varnothing\right) = \{(y, \varepsilon)\}$
  (agent $\mathcal{A}_2$ decides not to cut the cake: $\mathcal{A}_1$ is going to ensure it since it is her goal as recognizable by the existential quantifier in the first row).

Function $\mathfrak{C}$ provides a new parameter in the interpretation of the quantificational operators. Clause 5_2 is thus a schema that matches a variety of k-agent systems depending on the parameter $\mathfrak{C}$ and relationship $\models$ should have $\mathfrak{C}$ as index. Note that, due to the complexity of behaviors that function $\mathfrak{C}$ may encode, it is not possible to discharge quantificational operators in this semantics. The analogous of function τ_1 of Section 3.1 that is faithful for 5_2 may not exist.

Clause 5_2 is more general than 5_1. In particular, $\mathfrak{C}$ can formally capture cooperation. The collaboration among the agents is obtained by taking the whole set of variables of the collaborating agents as third argument for $\mathfrak{C}$. That is, $\mathfrak{C}$ provides the variable instantiations for all the agents at once by outputting a set $\{(x_1, \alpha_1), \ldots, (x_r, \alpha_r)\}$. We do not discuss the import of assumptions 1) and 2) further. Instead, below an alternative semantics is given since it sheds light on another issue.

3.3 Optimistic co-ordinated agents

In the discussion of Section 3.1, we mentioned two interpretations for the operators where both universal and existential quantifiers occur. One leads to clause 5_1. The other interpretation is rendered by a different function, here called τ_3, whose definition follows from that of τ_1 provided cases (5_1) and (6_1) are substituted by

$$\begin{bmatrix}\exists x\\ \forall y\end{bmatrix}\varphi \overset{\tau_3}{\longmapsto} \forall y \exists x \begin{bmatrix} x\\ y\end{bmatrix}\tau_3(\varphi) \qquad (5_3)$$

$$\begin{bmatrix}\forall x\\ \exists y\end{bmatrix}\varphi \overset{\tau_3}{\longmapsto} \forall x \exists y \begin{bmatrix} x\\ y\end{bmatrix}\tau_3(\varphi), \qquad (6_3)$$

respectively.

Here is the semantic clause that matches function τ_3:

5_3. Let $[X]$ be a quantificational operator with existentially quantified variables $x_1, \ldots, x_r$ and universally quantified variables $y_1, \ldots, y_s$ $(r, s \geq 0)$. Then
$\mathcal{M}, s, \Im \models [X]\varphi$ if: for all $\beta_1, \ldots, \beta_s \in$ Act, there exist $\alpha_1, \ldots, \alpha_r \in$ Act such that if Γ is the k-action obtained by substituting α_i for $\exists x_i$, β_j for $\forall y_j$, and $[\![c_h]\!]$ for each action identifier or free variable c_h in $[X]$ (for all relevant indices i, j, h), then for all $(s, t) \in R(\Gamma)$, $\mathcal{M}, t, \Im^* \models \varphi$ where $\Im^*(x_i) = \alpha_i$, $\Im^*(y_j) = \beta_j$, and $\Im^*(z) = \Im(z)$.

This clause captures the simple *possibility* for a given formula to be true. For instance, consider the paper/scissor/stone game. If p_0 stands for "$\mathcal{A}_1$ wins", then formula $\begin{bmatrix}\exists x\\ \forall y\end{bmatrix} p_0$ is valid according to this latter semantics and tells us that one has always a chance to win a play of this game. However, the same formula is false for 5_1 since that clause requires the existence of a winning strategy for each possible play.

As before, one can restate clause 5_3 using function $\mathfrak{C}$ explicitly. In this case the fifth argument of $\mathfrak{C}$ is crucial. This argument provides extra knowledge that the agents have while making their choices. In clause 5_3, the agents choosing for the existentially quantified variables are aware of the choices made for the universally quantified variables even though they are done concurrently. This information is provided by set $K = \bigcup_h \{(h, y_1, \beta_1), (h, y_2, \beta_2), \ldots, (h, y_s, \beta_s)\}$ where h ranges over the indeces of the agents associated to a universally quantified variable (such a generality allows us to directly extend the function to multi-column operators). Co-ordination is ensured by providing the whole set of variables $x_1, \ldots, x_r$ as third argument as we have seen in the reconstruction of 5_1 within clause 5_2.

We dub the agents satisfying 5_3 the *optimistic co-ordinated agents*.

We conclude this section with an observation. The difference between clause 5_1 and 5_3 corresponds to the difference between α-ability (effectiveness) and β-ability applied to coalitions (cf. [vHWo305] and the references therein). It is fairly easy to rewrite schema 5_2 and the conditions on parameter $\mathfrak{C}$ to capture β-ability.

4 Knowing the past, reasoning about the future

Consider a constant two-column[7] operator $\begin{bmatrix} c_1 & c_3 \\ c_2 & c_4 \end{bmatrix}$, call it $[C]$. From the definition of the valuation function and clause 4 of Section 2, multi-column constant operators split into simpler operators without loss of information. The following formula is valid

$$\begin{bmatrix} c_1 & c_3 \\ c_2 & c_4 \end{bmatrix} \varphi \equiv \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \begin{bmatrix} c_3 \\ c_4 \end{bmatrix} \varphi.$$

This equivalence does not hold for quantificational operators though, i.e. in general

$$[M_1 M_2]\varphi \not\equiv [M_1][M_2]\varphi \quad (M_1, M_2 \in \mathrm{QOP}). \tag{1.6}$$

One reason is that the order of instantiation of quantified variables may change in the two formulas. For instance, evaluating formula $\begin{bmatrix} \exists x_1 & \forall x_2 \\ \forall y_1 & \exists y_2 \end{bmatrix} \varphi$ with clause 5_1 we instantiate first variables x_1, y_2 and only later x_2 and y_1. The instantiation order for the same clause becomes x_1, y_1, y_2, x_2 when we

[7] We give examples using two-column operators. The generalization to n-column operators is often straightforward although some care is needed.

consider formula $\begin{bmatrix} \exists x_1 \\ \forall y_1 \end{bmatrix} \begin{bmatrix} \forall x_2 \\ \exists y_2 \end{bmatrix} \varphi$. This is not so for other semantic alternatives and one is free to adopt or reject a constraint like (1.6) by selecting an appropriate semantics.

It should be clear by now that to establish the truth value of a formula where the constant operator $[C]$ occurs, it is necessary to consider all the action identifiers (and their positions) occurring in $[C]$. For instance, knowing that c_1, c_3 are the actions executed by agent $\mathcal{A}_1$ (in that order) does not suffice to know which states are reachable.

Informally, the formula $\begin{bmatrix} c_1 & \exists x \\ c_2 & c_4 \end{bmatrix} p_0$ means: "first, agent $\mathcal{A}_1$ executes c_1 and (concurrently) agent $\mathcal{A}_2$ executes c_2, then agent $\mathcal{A}_1$ chooses and executes an action and (concurrently) agent $\mathcal{A}_2$ executes c_4". In the light of the previous section, one can interpret the existential quantifier in different ways. The set of choices for x will depend on what agent $\mathcal{A}_1$ knows about the formula itself and in particular about operator $\begin{bmatrix} c_1 & \exists x \\ c_2 & c_4 \end{bmatrix}$. For if she is aware of the presence of c_1, c_2, c_4 and of their positions, she can use the semantic clauses to verify if there is an action that executed after c_1, forces the system to states satisfying p_0. Agent $\mathcal{A}_1$ might rely on default rules (or preferences) when she lacks some information about the components of the operator.

To establish the truth value of the formula, it is important to state what agent $\mathcal{A}_1$ knows (or does not know) about the operator. Several options are possible. For instance, assuming perfect recall, one can assume that agent $\mathcal{A}_1$ is aware that c_1 is in position (1,1) of the modality marker since she has just executed that action. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are *totally isolated* agents, then one can assume that agent $\mathcal{A}_1$ has no information about what $\mathcal{A}_2$ has done earlier, that is, she has no knowledge on the content of position (2,1) of the operator. Analogously for $\mathcal{A}_2$. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are *isolated* but can observe each other's doings, at entry $(1, 2)$ agent $\mathcal{A}_1$ knows that c_2 is in position (2,1). For the simple reason that $\mathcal{A}_1$ and $\mathcal{A}_2$ act concurrently, agent $\mathcal{A}_1$ does know what $\mathcal{A}_2$ is going to execute as second action only if they are co-ordinating or action c_4 is public knowledge.[8] We conclude pointing out that also the quantifiers occurring in the operator may be hidden. After all, an agent may be aware or unaware of the *changes of attitude* in the other agents at different times including, perhaps, her own (past or future) changes.

[8] Most of these features are captured using the fifth argument of function $\mathfrak{C}$. Also, note that in Section 2 we implicitly assumed that the action identifiers in the quantificational operator are known to all the agents, they are *public knowledge*. This assumption is dropped here. Indeed, one may have a commitment to do a specific action c_i at some point and prevent other agents from knowing it or knowing when that action will take place. The semantic clauses we introduced can be modified to mirror these cases.

5 Modeling with quantificational operators

Our first example is in the area of planning. There are two agents, say Anthony ($\mathscr{A}_1$) and Bill ($\mathscr{A}_2$), and a project that must be finished by a certain time. Let us say that there are 3 time-steps before the deadline (step-1, step-2, and step-3) and that Anthony cannot work at the project at time-step 1 since at that time he has to meet his doctor. We use action identifier a for the action Anthony does at this step. Later, he is working full time on the project. Regarding Bill, he will work on the project except at the time-step 2 when he has to meet with the office manager. Bill does not know what the meeting is about. We represent this case in our language with the following formula (φ stands for "the project is finished"):

$$\begin{bmatrix} a & \exists x & \exists z \\ \exists y & \forall u & \exists v \end{bmatrix} \varphi \tag{1.7}$$

The first row describes Anthony's attitude toward the project during the three time-steps, while the second row describes Bill's attitude. Note that the universal quantifier marks the time-step when Bill acts without regards for the project since his action at that time depends on what his office manager asks him to do. If Anthony and Bill are risk-averse and co-operative agents, all the actions that instantiate variables x, z, y, v should be chosen together as described by clause 5_1, where we now allow $[X]$ to be a multi-column quantificational operator (the clause applies to multi-column operators without change). If the two agents work independently from each other (non co-operative agents), then we should adopt clause 5_3 (which also extends to multi-column operators).

We may want to model the case where the agents have a predefined plan for the first two time-steps only. For instance, suppose they agreed on a plan the day before when they knew they where going to be in different places during time-steps 1 and 2 without the possibility of sharing information. Also, let us assume that after time-step 2 they meet so that the decision about the third time-step can be postponed to that time. This situation is described by formula

$$\begin{bmatrix} a & \exists x \\ \exists y & \forall u \end{bmatrix} \begin{bmatrix} \exists z \\ \exists v \end{bmatrix} \varphi.$$

Note the change of the order in which quantified variables are instantiated: the value of u is known when choosing a value for z and v (Section 4).

The second example we consider comes from robotics. Here there are two agents whose goal is to pick up an object but none of them can do it alone. If φ stands for "the object is lifted", the situation is described by formula

$$\begin{bmatrix} \exists x \\ \exists y \end{bmatrix} \varphi \wedge \neg \begin{bmatrix} \forall x \\ \exists y \end{bmatrix} \varphi \wedge \neg \begin{bmatrix} \exists x \\ \forall y \end{bmatrix} \varphi. \tag{1.8}$$

Consider interpretation 5_1. If (1.8) is true, the agents can execute actions that make φ true (first conjunct) but the first or the second agent cannot bring about φ without the collaboration of the other agent (second and third conjuncts). It is possible to make a stronger claim adding the following as conjuncts: $\begin{bmatrix} \forall x \\ \exists y \end{bmatrix} \neg\varphi$, $\begin{bmatrix} \exists x \\ \forall y \end{bmatrix} \neg\varphi$. These tell us that each agent can force φ to be false, i.e., each can prevent the system from reaching any state where φ holds.

6 Related work and conclusions

We looked at the variety of semantics for quantificational operators extending the work in [$\text{Bo}_1$05a]. [$\text{Bo}_1$03] provides an interpretation for the quantificational modal operators that relies entirely on game-theoretic semantics. [$\text{Bo}_1$05b] studies the formal properties of an interpretation along the lines of 5_1. in the framework of standard Kripke semantics.

The formalism we adopted has been influenced by the notion of Henkin (branching) quantifiers [$\text{He}_1$61, $\text{Wa}_2$70]. Note that there is an ontological discrepancy between the notion of agent in multi-agent systems (where agents are internal components) and the formal notion of player as used in game-theory (players are external components that act to interpret the formalism); a distinction that has not received enough attention in the literature.

A modal version of Hintikka Independent-friendly logic [$\text{Hi}_1\text{Sa}_4$96], which comprises Henkin quantifiers, has been proposed in [$\text{Br}_0\text{Fr}_4$02b]. The aim of the authors is to isolate a notion of bisimulation (model equivalence) that corresponds to their modal system. Related to our work is also the logic **ATL** [$\text{Al}_2\text{He}_2\text{Ku}_1$02] and its extension **ATEL** [$\text{vHWo}_3$02]. The relationship is better analyzed through *Coalition Logic* (**CL**) introduced in [$\text{Pa}_7$02]. The connections between **CL** and **ATL** are presented in [$\text{Go}_2$01, $\text{Go}_2\text{Ja}_2$04]. Interestingly, the encoding of **CL** into our formalism enables the use of Kripke structures for **CL**. (More precisely, **CL** is semantically equivalent to a fragment of our logic with the semantics given by clause 5_1, cf. [$\text{Bo}_1$07]). Other frameworks, like **KARO** [$\text{vLvHMe}_3$98] and the variety of systems following the BDI approach [$\text{Ra}_3\text{Ge}_1$91] or the *Intention Logic* [$\text{Co}_1\text{Le}_1$90], adopt combinations of different modalities or exploit full first-order logic. These are very expressive systems and differ in their motivations from our approach. We refer the reader to [$\text{vHVe}_2$02, $\text{vHWo}_3$03, $\text{vDvHKo}_4$07] for overviews on this area of research.

We have shown how to produce different interpretations for modal operators built out of action identifiers, variables, and quantified variables. Our stand is that when there is a number of practical constraints to capture, semantic pluralism could help. We showed how the same language can distinguish and characterize different systems in a flexible way making it possible to describe uniformly what might seem a plethora of heterogeneous cases. Then, formal and reliable comparisons of apparently disparate phenomena become possible at the semantical level. Our approach has some drawbacks as well. The quantificational operators inherit some restrictions of Dynamic Logic, in particular the rigid structure in finite steps. (Extensions with constructs on action identifiers or temporal modalities have not been studied yet.) On the technical side, although adding quantificational modal operators does not make the resulting logic necessarily undecidable, this happens in many cases when equality (over action) is present. For instance, one can see that the theory in [Bo$_1$03] is undecidable by embedding first-order logic augmented with a binary predicate via $A(x,y) \longmapsto \begin{bmatrix} x \\ y \end{bmatrix} p_0$, for some atomic p_0. For an example in the opposite sense, [Bo$_1$05b] gives a complete and decidable logic for the class of multi-relational Kripke frames.

References

[Al$_0$Le$_0$04] J.J. Alferes & J.A. Leite, eds. *Logics in Artificial Intelligence: 9th European Conference, JELIA 2004*, *Lecture Notes in Computer Science* 3229. Springer, 2004.

[Al$_1$Fi$_0$Sa$_3$91] J.F. Allen, R. Fikes & E. Sandewall, eds. *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91).* Morgan Kaufmann, 1991.

[Al$_2$He$_2$Ku$_1$02] R. Alur, T. Henzinger & O. Kupferman. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713, 2002.

[Ba$_5$Ma$_3$05] S. Bandini & S. Manzoni, eds. *AI*IA 2005: Advances in Artificial Intelligence, 9th Congress of the Italian Association for Artificial Intelligence, Milan, Italy, September 21–23, 2005, Proceedings*, *Lecture Notes in Computer Science* 3673. Springer, 2005.

[Be$_0$+02] B. Bennett, C. Dixon, M. Fisher, U. Hustadt, E. Franconi, I. Horrocks & M. de Rijke. Combinations of modal logics. *Artificial Intelligence Review* 17(1):1–20, 2002.

[Bo$_1$03] S. Borgo. A multi-agent modal language for concurrency with non-communicating agents. In [Ma$_6$MüPe$_1$03, pp. 40–50].

[Bo$_1$05a] S. Borgo. Modal operators with adaptable semantics for multi-agent systems. In [Ba$_5$Ma$_3$05, pp. 186–197].

[$Bo_1$05b] S. Borgo. Quantificational modal logic with sequential Kripke semantics. *Journal of Applied Non-Classical Logics* 15(2):137–188, 2005.

[$Bo_1$07] S. Borgo. Coalitions in action logic. In [$Ve_0$07, pp. 1822–1827].

[$Br_0Fr_4$02b] J.C. Bradfield & S.B. Fröschle. On logical and concurrent equivalences. *Electronic Notes in Theoretical Computer Science* 52(1):32–45, 2002.

[Br_2He_3Tr05] J. Broersen, A. Herzig & N. Troquard. From coalition logic to STIT. *Electronic Notes in Theoretical Computer Science* 157(4):23–35, 2005.

[$Ca_4Jo_1$02] C. Castelfranchi & W. Johnson, eds. *First International Joint Conference on Autonomous Agents and Multiagent Systems.* ACM Press, 2002.

[$Co_1Le_1$90] P.R. Cohen & H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence* 42(2–3):213–261, 1990.

[$Go_2$01] V. Goranko. Coalition games and alternating temporal logics. In [vB01d, pp. 259–272].

[$Go_2Ja_2$04] V. Goranko & W. Jamroga. Comparing semantics of logics for multi-agent systems. *Synthese* 139(2):241–280, 2004.

[$Ha_1Ko_6Ti_1$00] D. Harel, D. Kozen & J. Tiuryn. *Dynamic Logic.* The MIT Press, 2000.

[$He_1$61] L. Henkin. Some remarks on infinitely long formulas. In *Infinitistic Methods: Proceedings of the Symposium on Foundations of Mathematics, Warsaw, 2–9 September 1959*, pp. 167–183. Pergamon Press and Państwowe Wydawnictwo Naukowe, 1961.

[$Hi_1Sa_4$96] J. Hintikka & G. Sandu. Game-theoretical semantics. In [vBtM96, pp. 361–410].

[$Ho_6$01] J. Horty. *Agency and Deontic Logic.* Oxford University Press, 2001.

[$Kr_2Mo_4$95] M. Krynicki & M. Mostowski. Henkin quantifiers. In [Kr_2Mo_4Sz95, pp. 193–262].

[$\text{Kr}_2\text{Mo}_4\text{Sz}95$] M. Krynicki, M. Mostowski & L.W. Szczerba, eds. *Quantifiers: Logics, Models and Computation.* Kluwer Academic Publishers, 1995.

[$\text{Ma}_6\text{MüPe}_1 03$] V. Mařík, J.P. Müller & M. Pechoucek, eds. *Multi-Agent Systems and Applications III, 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, Prague, Czech Republic, June 16–18, 2003, Proceedings*, *Lecture Notes in Computer Science* 2691. Springer, 2003.

[$\text{Pa}_7 02$] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation* 12(1):149–166, 2002.

[$\text{Pe}_4\text{Ma}_9 02$] L. Petrosjan & V. Mazalov, eds. *Game Theory and Applications.* Nova Science Publishers, 2002.

[$\text{Ra}_3\text{Ge}_1 91$] A. Rao & M. Georgeff. Modeling rational agents within a BDI-architecture. In [$\text{Al}_1\text{Fi}_0\text{Sa}_3 91$, pp. 473–484].

[vB01d] J. van Benthem, ed. *Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge (TARK VIII).* Morgan Kaufmann, 2001.

[vBtM96] J. van Benthem & A. ter Meulen, eds. *Handbook of Logic and Language.* Elsevier, 1996.

[$\text{vHVe}_2 02$] W. van der Hoek & R. Verbrugge. Epistemic logic: a survey. In [$\text{Pe}_4\text{Ma}_9 02$, pp. 53–94].

[$\text{vHWo}_3 02$] W. van der Hoek & M.J. Wooldridge. Tractable multiagent planning for epistemic goals. In [$\text{Ca}_4\text{Jo}_1 02$, pp. 1167–1174].

[$\text{vHWo}_3 03$] W. van der Hoek & M.J. Wooldridge. Towards a logic of rational agency. *Logic Journal of the IGPL* 11(2):135–159, 2003.

[$\text{vHWo}_3 05$] W. van der Hoek & M.J. Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence* 164(1–2):81–119, 2005.

[$\text{vDvHKo}_4 07$] H. van Ditmarsch, W. van der Hoek & B. Kooi. *Dynamic Epistemic Logic*, *Synthese Library* 337. Springer, 2007.

[$\text{vLvHMe}_3 98$] W. van Linder, W. van der Hoek & J.-J.C. Meyer. Formalizing abilities and opportunities of agents. *Fundamenta Informaticae* 34(1–2):53–101, 1998.

[Ve$_0$07] M.M. Veloso, ed. *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6–12, 2007.* 2007.

[Wa$_2$70] W. Walkoe. Finite partially-ordered quantification. *Journal of Symbolic Logic* 35(4):535–555, 1970.

[We$_1$99] G. Weiss, ed. *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence.* The MIT Press, 1999.

[Wö04] S. Wölfl. Qualitative action theory: A comparison of the semantics of Alternating-Time Temporal Logic and the Kutschera-Belnap approach to agency. In [Al$_0$Le$_0$04, pp. 70–81].

A Note on Kuhn's Theorem[*]

Adam Brandenburger

Stern School of Business
New York University
44 West Fourth Street
New York, NY 10012, United States of America
adam.brandenburger@stern.nyu.edu

Abstract

We revisit Kuhn's classic theorem on mixed and behavior strategies in games. We frame Kuhn's work in terms of two questions in decision theory: What is the relationship between global and local assessment of uncertainty? What is the relationship between global and local optimality of strategies?

This note is a homage to Kuhn's classic theorem on the replacement of mixed by behavior strategies in games [$\text{Ku}_0$50, $\text{Ku}_0$53]. It reframes Kuhn's work as two results in decision theory—i.e., in the context of trees involving a decision maker and Nature. The motivation is to see the meaning of Kuhn's work at this basic level.

The decision-theoretic framing in this note is in accordance with the so-called epistemic approach to game theory. Under the epistemic approach, a game is a multi-player decision problem—more exactly, a collection of decision problems, one for each player. In line with decision theory, a player is assumed to form a (subjective) probability assessment over the strategies chosen by other players in the game, and to choose an optimal strategy under this assessment. The questions à la Kuhn are then: (a) the relationship between global and local assessments; and (b) the relationship between global and local optimality.

The epistemic approach is 'the other way round' from the traditional approach to game theory. Under the traditional approach, we talk about a mixed strategy of a player, not another player's global assessment of the first player's deterministic choice of (pure) strategy. Likewise, we talk about

* My thanks to Scott Borg, Amanda Friedenberg, Rena Henderson, Konrad Grabiszewski, Mike Moldoveanu, Ben Polak, John Pratt, Nikolaj Siggelkow, Pat Sileo, and Gus Stuart for important input. A referee made very helpful suggestions. Financial support from Harvard Business School and the Stern School of Business is gratefully acknowledged.

a behavioral strategy of a player, not another player's system of local assessments about the first player. The mixed-behavioral framing is Kuhn's, of course.

In Section 5, we expand on the significance for Kuhn's Theorem of taking an epistemic perspective on games.

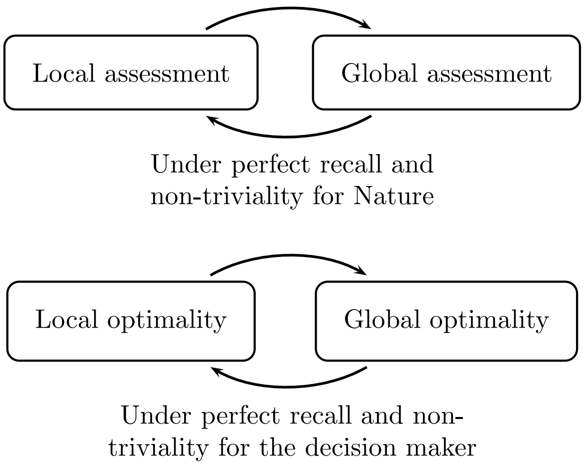

Local assessment
Global assessment
Under perfect recall and non-triviality for Nature
Local optimality
Global optimality
Under perfect recall and non-triviality for the decision maker

FIGURE 1. Summary of results

Figure 1 is a summary of the two results we cover. Each result is in two parts. For the first, we have: (i) Given a system of local probability assessments by the decision maker, i.e., an assessment over Nature's moves at each of Nature's information sets, there is a global assessment over Nature's strategies ("states") that yields the same probability of each path through the tree. (ii) If Nature has perfect recall and all chance nodes are non-trivial, then, given a global assessment by the decision maker, there is an equivalent system of local assessments. For the second result, we have: (i) If a strategy of the decision maker is locally optimal, i.e., optimal at each information set of the decision maker, then it is globally ("ex ante") optimal. (ii) If the decision maker has perfect recall and all decision nodes are non-trivial, then, if a strategy is globally optimal, it is locally optimal.

There is also a sufficiency result (which follows from part (ii) of the first result): Assume perfect recall and non-triviality for Nature. Then, it is enough to know the system of local assessments associated with any global assessment, to know which strategies are globally optimal. Putting this together with part (ii) of the second result gives: Assume perfect recall and non-triviality for both the decision maker and Nature. Then, to determine if a strategy is locally optimal, it is enough to know the system of local assessments of the decision maker.

We acknowledge that much (all?) of the contents of this note may be well known. Still, we hope that a self-contained presentation will be useful.

1 Decision Trees

A decision tree will be a two-person game in extensive form, where one player is the decision maker and the other is Nature. We now give formal definitions following Kuhn [$\mathrm{Ku}_0$50, $\mathrm{Ku}_0$53] (also the presentation in Hart [$\mathrm{Ha}_4$92]).

Definition 1.1. A (**finite**) **decision tree** consists of:

(a) A set of two players, one called the **decision maker** and the other called **Nature**.

(b) A finite rooted tree.

(c) A partition of the set of non-terminal nodes of the tree into two subsets denoted N (with typical element n) and M (with typical element m). The members of N are called **decision nodes**, and the members of M are called **chance nodes**.

(d) A partition of N (resp. M) into **information sets** denoted I (resp. J) such that for each I (resp. J):

   (i) all nodes in I (resp. J) have the same number of outgoing branches, and there is a given 1-1 correspondence between the sets of outgoing branches of different nodes in I (resp. J);

   (ii) every path in the tree from the root to a terminal node crosses each I (resp. J) at most once.

Note: The focus of the well-known literature on the "paradox of the absent-minded driver" (Piccione and Rubinstein [$\mathrm{Pi}_0\mathrm{Ru}_1$97]) is on non-Kuhn trees—specifically, trees that fail condition (d.ii) above. (See also Isbell [Is57].) We consider only Kuhn trees.

For each information set I (resp. J), number the branches going out of each node in I (resp. J) from 1 through $\#I$ (resp. $\#J$) so that the 1-1 correspondence in (d.i) above is preserved.

Definition 1.2. A **strategy** (of the decision maker) associates with each information set I, an integer between 1 and $\#I$, to be called the **choice** of the decision maker at I. Let S denote the set of strategies of the decision maker. A **state of the world** (or **state**) associates with each information set J, an integer between 1 and $\#J$, to be called the **choice** of Nature at J. Let Ω denote the set of states.

Note that a pair (s, ω) in $S \times \Omega$ induces a unique path through the tree.

Definition 1.3. Fix a path p through the tree and a strategy s. Say p is **allowed under** s if there is a state ω such that (s, ω) induces p.

Definition 1.4. Fix a path p through the tree and a state ω. Say p is **allowed under** ω if there is a strategy s such that (s, ω) induces p.

Definition 1.5. Fix a node n in N and a strategy s. Say n is **allowed under** s if there is a state ω such that the path induced by (s, ω) passes through n. Say an information set I is **allowed under** s if some n in I is allowed under s.

Definition 1.6. Say the decision maker has **perfect recall** if for any strategy s, information set I, and nodes n and n^* in I, node n is allowed under s if and only if node n^* is allowed under s.

Definition 1.7. Say a node n in N is **non-trivial** if it has at least two outgoing branches.

Definition 1.8. Fix a node m in M and a state ω. Say m is **allowed under** ω if there is a strategy s such that the path induced by (s, ω) passes through m. Say an information set J is **allowed under** ω if some m in J is allowed under ω.

Definition 1.9. Say Nature has **perfect recall** if for any state ω, information set J, and nodes m and m^* in J, node m is allowed under ω if and only if node m^* is allowed under ω.

Definition 1.10. Say a node m in M is **non-trivial** if it has at least two outgoing branches.

Example 1.11. Figure 2 depicts a case of imperfect recall for the decision maker. (The circular node belongs to Nature and the square nodes belong to the decision maker.) Let s be the strategy that chooses B at information set I (and b, say, at information set I'). Then node n is allowed under s but node n^* is not.

Example 1.12. Figure 3 is the standard example of imperfect recall for Nature. Let ω be the state that chooses U at information set J (and u, say, at information set J'). Then node m is allowed under ω but node m^* is not.

Define a relation of precedence on information sets I of the decision maker, as follows: Given two information sets I and I', say that I **precedes** I' if there are nodes n in I and n' in I' such that the path from the

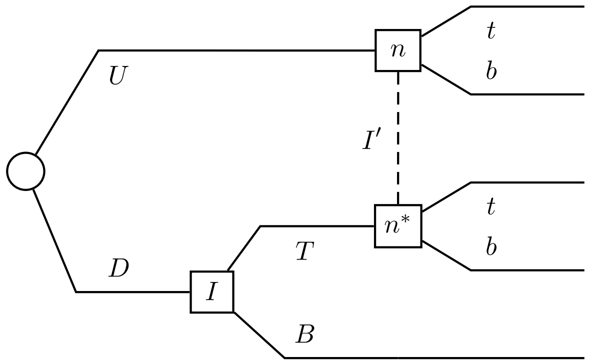


FIGURE 2.

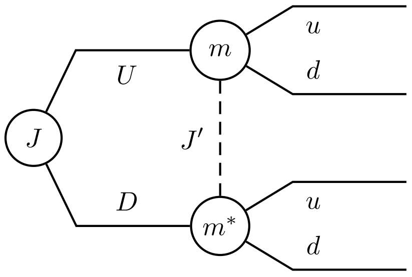


FIGURE 3.

root to n' passes through n. It is well known that if the decision maker has perfect recall and all decision nodes are non-trivial, then this relation is irreflexive and transitive, and each information set I has at most one immediate predecessor. (Proofs of these assertions can be constructed from arguments in Wilson [Wi$_0$72]. See also the appendix to this note.) Of course, the parallel statements hold for Nature.

In [Ku$_0$53], Kuhn observes that perfect recall implies that a player remembers: (i) all of his choices at previous nodes; and (ii) everything he knew at those nodes. The following two lemmas formalize these observations. (The proofs are in the appendix.) Again, parallel statements hold for Nature. (Lemma 1.13 will be used later.)

Lemma 1.13. Suppose the decision maker has perfect recall and all decision nodes are non-trivial. Fix information sets I and I', and strategies s

and s'. Suppose that I' is allowed under both s and s', and I precedes I'. Then I is allowed under both s and s', and s and s' coincide at I.

Continuation of Example 1.11. Let s choose T at I, and s' choose B at I. Then I' is allowed under both s and s', as is I. But s and s' differ at I. So Lemma 1.13 fails without perfect recall. In words, the decision maker forgets at I' whether he chooses T or B at I.

Next, write

$$[I] = \{\omega : I \text{ is allowed under } \omega\}.$$

Lemma 1.14. Suppose the decision maker has perfect recall and all decision nodes are non-trivial. Fix information sets I and I'. If I' succeeds I, then $[I'] \subseteq [I]$.

Continuation of Example 1.11. We have $[I] = \{D\}$ and $[I'] = \{U, D\}$. So Lemma 1.14, too, fails without perfect recall. In words, the decision maker knows at I that Nature doesn't choose U, but forgets this at I'.

A brief comment on the literature on these 'structural' aspects of perfect recall. Bonanno makes a nice distinction between "action" and "choice" [Bo$_0$04]. (The same action can be taken at different information sets.) He offers definitions of "knowing the actions you previously took" and "knowing what you previously knew", and shows that together these conditions characterize perfect recall. Ritzberger provides several characterizations of perfect recall [Ri$_0$99]. Van Benthem studies game trees as structures for logical languages, and, in particular, provides a dynamic epistemic logic-based axiomatization of games satisfying perfect-recall like conditions [vB01a]. Also related are the temporal logics in Halpern, van der Meyden, and Vardi [Ha$_0$vMVa04]. See [Bo$_0$04, Section 6] for further discussion of the literature.

2 Global and Local Probabilities

We now define the global and local probabilities on the tree, and then state Kuhn's Theorem.

Definition 2.1. A **global** probability measure on the tree is a probability measure on the set of states Ω.

Definition 2.2. A system of **local** probability measures on the tree associates with each information set J of Nature, a probability measure on the set of choices at J.

Fix a global measure σ, and a system of local measures $\pi(\cdot; J)$. Fix also a path p through the tree. Let J_1 be the first information set of Nature crossed by p, and let j_1 be the choice at J_1 that lies on p. Define $J_2, j_2, \ldots, J_K, j_K$

similarly, where J_K is the last information set of Nature crossed by p, and j_K is the choice at J_K that lies on p. (Note this is well defined, by condition (d.ii) of Definition 1.1. Also, we don't indicate the dependence of the indices $1, \ldots, K$ on p, but no confusion should result.)

Definition 2.3. The **global** probability of p is

$$\lambda(p; \sigma) = \sigma(\{\omega : p \text{ is allowed under } \omega\}).$$

Definition 2.4. The **local** probability of p is

$$\mu(p; \pi(\cdot; J_1), \ldots, \pi(\cdot; J_K)) = \prod_{k=1}^{K} \pi(j_k; J_k).$$

Continuation of Example 1.12. To practice these definitions, let σ assign probability 1/2 to (U, u) and probability 1/2 to (D, d). Also, let $\pi(U; J) = \pi(D; J) = 1/2$, and $\pi(u; J') = \pi(d; J') = 1/2$. Suppose p is the path induced by (U, u). Then the global probability of p is $\lambda(p; \sigma) = 1/2$, and the local probability of p is $\mu(p; \pi(\cdot; J), \pi(\cdot; J')) = 1/2 \times 1/2 = 1/4$.

We now state Kuhn's Theorem in the form of the following two results, and give proofs in the notation of this note.

Theorem 2.5. Fix a system of local measures $\pi(\cdot; J)$. There is a global measure σ such that for any path p,

$$\lambda(p; \sigma) = \mu(p; \pi(\cdot; J_1), \ldots, \pi(\cdot; J_K)).$$

Proof. It will be convenient to write Ω as a product space $\prod_J \boldsymbol{C}(J)$, where $\boldsymbol{C}(J)$ denotes the set of choices at information set J. Given a state ω, write $\omega(J)$ for the Jth coordinate of ω, i.e., the choice ω makes at J. Set $\sigma(\omega) = \prod_J \pi(\omega(J); J)$. This is readily seen to define a probability measure on Ω.

Now fix a path p, and let $J_1, j_1, \ldots, J_K, j_K$ be defined as earlier. For $k = 1, \ldots, K$, let

$$A_k = \{\omega : \omega(J_k) = j_k\}.$$

Note that $\sigma(A_k) = \pi(j_k; J_k)$. The set of ω such that p is allowed under ω is $\bigcap_{k=1}^{K} A_k$, and, since σ is a product measure,

$$\sigma\Big(\bigcap_{k=1}^{K} A_k\Big) = \prod_{k=1}^{K} \sigma(A_k) = \prod_{k=1}^{K} \pi(j_k; J_k),$$

as required. Q.E.D.

We need one more definition, used in the proof of the next result.

Definition 2.6. Fix a global measure σ and an information set J. The **localized** probability measure at J is given by

$$\pi(j; J, \sigma) = \frac{\sigma(\{\omega : J \text{ is allowed under } \omega, \text{ and } \omega(J) = j\})}{\sigma(\{\omega : J \text{ is allowed under } \omega\})},$$

if $\sigma(\{\omega : J \text{ is allowed under } \omega\}) > 0$. If $\sigma(\{\omega : J \text{ is allowed under } \omega\}) = 0$, define $\pi(\cdot; J, \sigma)$ arbitrarily.

Theorem 2.7. Suppose Nature has perfect recall and all chance nodes are non-trivial. Fix a global measure σ. There is a system of local measures $\pi(\cdot; J)$ such that for any path p,

$$\mu(p; \pi(\cdot; J_1), \ldots, \pi(\cdot; J_K)) = \lambda(p; \sigma).$$

Proof. Fix a path p and $J_1, j_1, \ldots, J_K, j_K$ as earlier. Let

$$\begin{aligned} B &= \{\omega : p \text{ is allowed under } \omega\}, \\ C_k &= \{\omega : J_k \text{ is allowed under } \omega, \text{ and } \omega(J_k) = j_k\}, \\ D_k &= \{\omega : J_k \text{ is allowed under } \omega\}, \end{aligned}$$

for $k = 1, \ldots, K$.

We show that for $k = 1, \ldots, K-1$, $C_k \subseteq D_{k+1}$. Suppose p passes through node m in J_k. Fix ω such that J_k is allowed under ω. Then by perfect recall, node m is allowed under ω. That is, there is a strategy s such that the path induced by (s, ω) passes through m. Let (s', ω') induce the path p. Then s and s' coincide at all information sets of the decision maker crossed by p from the root to m. Indeed, we can take $s = s'$. We know that $\omega'(J_k) = j_k$. Therefore, if $\omega(J_k) = j_k$, the path induced by (s', ω) coincides with the path induced by (s', ω'), from the root to J_{k+1}. Certainly then, J_{k+1} is allowed under ω, as required.

We next show that for $k = 1, \ldots, K-1$, $D_{k+1} \subseteq C_k$. Let (s', ω') be as above, so that certainly J_{k+1} is allowed under ω'. Fix ω such that J_{k+1} is allowed under ω. Since J_k precedes J_{k+1}, Lemma 1.13 (stated for Nature) implies that: (i) J_k is allowed under ω; and (ii) ω and ω' coincide at J_k, i.e., $\omega(J_k) = j_k$.

We now have that for $k = 1, \ldots, K-1$, $C_k = D_{k+1}$. By definition, $C_k \subseteq D_k$ for each k. This shows that the C_k's are a decreasing sequence.

Given a global measure σ, define a system of local measures $\pi(\cdot; J)$ by setting $\pi(\cdot; J) = \pi(\cdot; J, \sigma)$.

Note that $C_K = B$, since J_K is the last information set of Nature crossed by p. It follows that if $\lambda(p; \sigma) = \sigma(B) > 0$, then $\sigma(C_k) > 0$ and $\sigma(D_k) > 0$

for each k. We then have

$$\mu(p; \pi(\cdot; J_1), \ldots, \pi(\cdot; J_K)) = \prod_{k=1}^{K} \frac{\sigma(C_k)}{\sigma(D_k)}. \tag{2.1}$$

But the numerator of each term in (2.1) cancels with the denominator of the next term, leaving

$$\mu(p; \pi(\cdot; J_1), \ldots, \pi(\cdot; J_K)) = \frac{\sigma(C_K)}{\sigma(D_1)}.$$

We already have $\sigma(C_K) = \sigma(B)$. Also, $D_1 = \Omega$, since J_1 is the first information set of Nature crossed by p, so $\sigma(D_1) = 1$. This establishes that

$$\mu(p; \pi(\cdot; J_1), \ldots, \pi(\cdot; J_K)) = \sigma(B) = \lambda(p; \sigma), \tag{2.2}$$

as required.

Now suppose $\lambda(p; \sigma) = \sigma(B) = 0$. If $\sigma(D_k) > 0$ for each k, we still get (2.1), from which we get (2.2), and so $\mu(p; \pi(\cdot; J_1), \ldots, \pi(\cdot; J_K)) = 0$, as required. We have $\sigma(D_1) = 1$, so the remaining case is that $\sigma(D_k) = 0$ for some $k = 2, \ldots, K$. Choose the minimum such k. Then $\sigma(D_{k-1}) > 0$. Also $\sigma(C_{k-1}) = 0$, since $C_{k-1} = D_k$. Thus $\pi(j_{k-1}; J_{k-1}) = \sigma(C_{k-1})/\sigma(D_{k-1}) = 0$, so that $\mu(p; \pi(\cdot; J_1), \ldots, \pi(\cdot; J_K)) = 0$, as required. Q.E.D.

Continuation of Example 1.12. Theorem 2.7 fails without perfect recall. To see this, let σ assign probability 1/2 to (U, u) and probability 1/2 to (D, d), as before. Then, in particular, we need $\pi(U; J) \times \pi(u; J') = 1/2$, $\pi(U; J) \times \pi(d; J') = 0$, and $\pi(D; J) \times \pi(d; J') = 1/2$, which is impossible.

3 Global and Local Optimality

Next we define global and local optimality of a strategy of the decision maker.

Definition 3.1. A **payoff function** (of the decision maker) is a map $V : S \times \Omega \to \mathbb{R}$ satisfying $V(s, \omega) = V(s', \omega')$ whenever (s, ω) and (s', ω') induce the same path.

Definition 3.2. Fix a probability measure σ on Ω. A strategy s is **globally optimal** under σ if

$$\sum_{\omega \in \Omega} \sigma(\omega) V(s, \omega) \geq \sum_{\omega \in \Omega} \sigma(\omega) V(r, \omega)$$

for every other strategy $r \in S$.

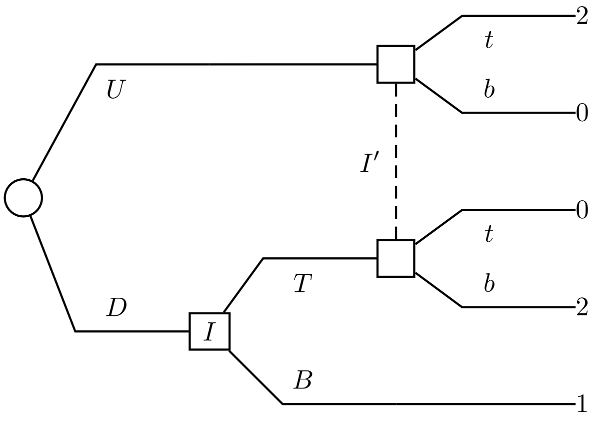


FIGURE 4.

Definition 3.3. Fix a probability measure σ on Ω. Fix also a strategy s and an information set I that is allowed under s and satisfies $\sigma([I]) > 0$. Then s is **locally optimal at** I under σ if

$$\sum_{\omega\in\Omega} \sigma(\omega|[I])V(s,\omega) \geq \sum_{\omega\in\Omega} \sigma(\omega|[I])V(r,\omega)$$

for every other strategy $r \in S$ under which I is allowed. Strategy s is **locally optimal** under σ if for every information set I that is allowed under s and satisfies $\sigma([I]) > 0$, it is locally optimal at I under σ.

In words, a strategy is globally optimal if it is expected-payoff maximizing under the (unconditional) measure σ. It is locally optimal if it is expected-payoff maximizing under each conditional measure $\sigma(\omega|[I])$ that is defined (and where I is allowed under the strategy).

Example 3.4. Figure 4 is Figure 2 with payoffs added for the decision maker. Let σ assign probability 2/3 to U and 1/3 to D. Then Tt, Bt, Tb, and Bb yield expected payoffs of 4/3, 5/3, 2/3, and 1/3, respectively—so Bt is (uniquely) globally optimal under σ.

As noted before, $[I] = \{D\}$ and $[I'] = \{U, D\} = \Omega$. So, $\sigma([I]) > 0$ and $\sigma([I']) = 1$. Also, both I and I' are allowed under all four strategies. It follows that local optimality at I' is the same as global optimality. At I, we find that Tt, Bt, Tb, and Bb yield conditional expected payoffs of 0, 1, 2, and 1, respectively—so, in fact, no strategy is locally optimal under σ.

Theorem 3.5. Fix a probability measure σ on Ω. If a strategy s is locally optimal under σ, then it is globally optimal under σ.

Proof. Partition Ω into cells $I_0, I_1, \dots, I_L$, where $\omega \in I_\ell$, for some $\ell = 1, \dots, L$, if I_ℓ is the first information set of the decision maker allowed under ω, and $\omega \in I_0$ if there is no information set of the decision maker allowed under ω.

Write

$$\sum_{\omega\in\Omega} \sigma(\omega)V(s,\omega) = \sum_{\ell=0}^{L} \sigma([I_\ell]) \sum_{\omega\in\Omega} \sigma(\omega|[I_\ell])V(s,\omega),$$

where we take $\sigma(\cdot|[I_\ell])$ to be arbitrary if $\sigma([I_\ell]) = 0$. Suppose s is locally optimal under σ, but not globally optimal under σ. Then there must be another strategy r such that

$$\sum_{\omega\in\Omega} \sigma(\omega)V(r,\omega) > \sum_{\omega\in\Omega} \sigma(\omega)V(s,\omega).$$

But

$$\sum_{\omega\in\Omega} \sigma(\omega)V(r,\omega) = \sum_{\ell=0}^{L} \sigma([I_\ell]) \sum_{\omega\in\Omega} \sigma(\omega|[I_\ell])V(r,\omega),$$

so there must be $\ell = 0, \dots, L$ such that $\sigma([I_\ell]) > 0$ and

$$\sum_{\omega\in\Omega} \sigma(\omega|[I_\ell])V(r,\omega) > \sum_{\omega\in\Omega} \sigma(\omega|[I_\ell])V(s,\omega).$$

Note that in fact $1 \le \ell \le L$, since, on I_0, $V(\cdot,\omega)$ is independent of the strategy. Since it is a first information set of the decision maker, I_ℓ must be allowed under r. This implies that s is not locally optimal under σ at I_ℓ, a contradiction. Q.E.D.

Theorem 3.6. Suppose the decision maker has perfect recall and all decision nodes are non-trivial. Fix a probability measure σ on Ω. If a strategy s is globally optimal under σ, then it is locally optimal under σ.

Notes: (i) By finiteness, a globally optimal strategy always exists under any σ. So Theorem 3.6 implies, in particular, that under the given conditions a locally optimal strategy also exists under any σ. (ii) Kline [Kl$_1$02] contains a stronger result, based on a weakening of perfect recall.

Proof. Suppose that s is globally optimal and that, contra hypothesis, there is an information set I allowed under s and satisfying $\sigma([I]) > 0$, such that

$$\sum_{\omega\in\Omega} \sigma(\omega|[I])V(s,\omega) < \sum_{\omega\in\Omega} \sigma(\omega|[I])V(r,\omega) \tag{3.1}$$

for some other strategy $r \in S$ under which I is allowed.

Construct the strategy q that coincides with r at I and all succeeding information sets of the decision maker, and coincides with s elsewhere.

We first show that if $\omega \in [I]$, then $V(q, \omega) = V(r, \omega)$.

From $\omega \in [I]$, there is a node n_1 in I and a strategy s_1 such that the path induced by (s_1, ω) passes through n_1. Since I is allowed under s, there is a node n_2 in I and a state ω_2 such that the path induced by (s, ω_2) passes through n_2. By perfect recall, there is then a state ω_3 such that the path induced by (s, ω_3) passes through n_1. Now consider the information sets I' crossed by the path from the root to n_1. Since the paths induced by (s_1, ω) and (s, ω_3) both pass through n_1, the strategies s_1 and s must coincide at these sets. Similarly, consider the information sets J crossed by the path from the root to n_1. Since the paths induced by (s_1, ω) and (s, ω_3) both pass through n_1, the states ω and ω_3 must coincide at these sets. Therefore, the path induced by (s, ω) must pass through n_1.

We can repeat the argument with strategy r in place of strategy s, to conclude that the path induced by (r, ω) must also pass through n_1. But then, using the definition of strategy q, the paths induced by (q, ω) and (r, ω) must be the same. Thus $V(q, \omega) = V(r, \omega)$, as required.

Next, we show that if $\omega \in \Omega \backslash [I]$, then $V(q, \omega) = V(s, \omega)$. From $\omega \in \Omega \backslash [I]$, the path induced by (s, ω) does not cross I, and therefore does not cross any information set of the decision maker that succeeds I. Consider the information sets I' that are in fact crossed by the path induced by (s, ω). By construction, the strategies q and s coincide at each such I'. Thus the paths induced by (q, ω) and (s, ω) are the same, and so $V(q, \omega) = V(s, \omega)$, as required. Write

$$\begin{aligned}
&\sum_{\omega \in \Omega} \sigma(\omega) V(q, \omega) \\
&\qquad = \sigma([I]) \sum_{\omega \in \Omega} \sigma(\omega | [I]) V(q, \omega) + \sigma(\Omega \backslash [I]) \sum_{\omega \in \Omega} \sigma(\omega | \Omega \backslash [I]) V(q, \omega),
\end{aligned}$$

where $\sigma(\cdot | \Omega \backslash [I])$ is arbitrary if $\sigma(\Omega \backslash [I]) = 0$. We have

$$\begin{aligned}
&\sum_{\omega \in \Omega} \sigma(\omega) V(q, \omega) \\
&\qquad = \sigma([I]) \sum_{\omega \in \Omega} \sigma(\omega | [I]) V(r, \omega) + \sigma(\Omega \backslash [I]) \sum_{\omega \in \Omega} \sigma(\omega | \Omega \backslash [I]) V(s, \omega) \\
&\qquad > \sigma([I]) \sum_{\omega \in \Omega} \sigma(\omega | [I]) V(s, \omega) + \sigma(\Omega \backslash [I]) \sum_{\omega \in \Omega} \sigma(\omega | \Omega \backslash [I]) V(s, \omega) \\
&\qquad = \sum_{\omega \in \Omega} \sigma(\omega) V(s, \omega),
\end{aligned}$$

where the inequality uses (3.1) and $\sigma([I]) > 0$. But this contradicts the global optimality of s. Q.E.D.

Continuation of Example 3.4. Theorem 3.6 fails without perfect recall (for the decision maker). Indeed, we saw that only Bt is globally optimal, but it is not locally optimal at I.

4 A Sufficiency Result

We now establish a sufficiency result: Assume perfect recall and non-triviality for Nature. Then, it is enough to know the localized probabilities associated with any probability measure, to know which strategies are globally optimal.

First some notation. As in Section 2, given a path p, write J_1 for the first information set of Nature crossed by p,..., J_K for the last information set of Nature crossed by p (and suppress the dependence on p). Also, in this section it will be helpful to write $W(p)$ for the payoff $V(s,\omega)$, if (s,ω) induces the path p.

Definition 4.1. Fix two (global) probability measures σ and τ on Ω. Say σ and τ are **locally equivalent** if for each path p, $\lambda(p;\sigma) > 0$ if and only if $\lambda(p;\tau) > 0$, and, in this case,

$$\mu(p;\pi(\cdot;J_1,\sigma),\ldots,\pi(\cdot;J_K,\sigma)) = \mu(p;\pi(\cdot;J_1,\tau),\ldots,\pi(\cdot;J_K,\tau)).$$

In words, two measures are locally equivalent if they give rise to the same localized probability of each path that gets positive (global) probability.

Example 4.2. In the tree in Figure 5, let σ assign probability 1/2 to (U,u) and probability 1/2 to (D,d), and τ assign probability 1/4 to each of (U,u), (U,d), (D,u), and (D,d). It can be checked that σ and τ are locally equivalent.

Here is the sufficiency result:

Theorem 4.3. Suppose Nature has perfect recall and all chance nodes are non-trivial. Let σ and τ be probability measures on Ω that are locally equivalent. Then for any strategy s,

$$\sum_{\omega\in\Omega}\sigma(\omega)V(s,\omega) = \sum_{\omega\in\Omega}\tau(\omega)V(s,\omega).$$

Proof. Write

$$\sum_{\omega\in\Omega}\sigma(\omega)V(s,\omega) = \sum_{\{p:\, p \text{ is allowed under } s\}}\ \sum_{\{\omega:\, (s,\omega) \text{ induces } p\}}\sigma(\omega)W(p). \quad (4.1)$$

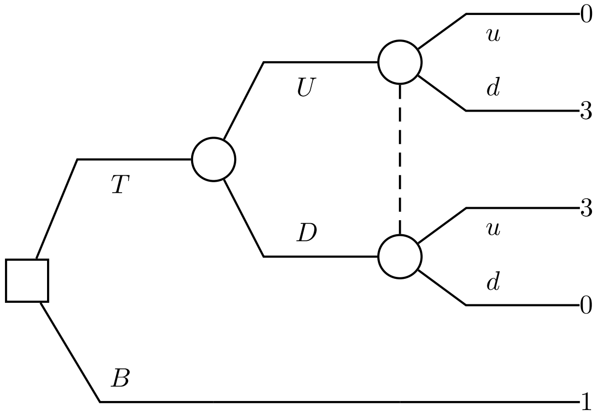


Figure 5.

Now, if (s, ω) induces p, then certainly p is allowed under ω. Conversely, suppose p is allowed under ω. That is, there is an s' such that (s', ω) induces p. Suppose also that p is allowed under s. That is, there is an ω' such that (s, ω') induces p. It follows (by arguing forwards along the path p) that (s, ω) must also induce p. Using the definition of $\lambda(p; \sigma)$, this establishes that (4.1) can be rewritten as

$$\sum_{\omega \in \Omega} \sigma(\omega) V(s, \omega) = \sum_{\{p: p \text{ is allowed under } s\}} \lambda(p; \sigma) W(p). \tag{4.2}$$

By the same argument we can write

$$\sum_{\omega \in \Omega} \tau(\omega) V(s, \omega) = \sum_{\{p: p \text{ is allowed under } s\}} \lambda(p; \tau) W(p). \tag{4.3}$$

Fix a path p. By the proof of Theorem 2.7,

$$\lambda(p; \sigma) = \mu(p; \pi(\cdot; J_1, \sigma), \ldots, \pi(\cdot; J_K, \sigma)), \tag{4.4}$$

$$\lambda(p; \tau) = \mu(p; \pi(\cdot; J_1, \tau), \ldots, \pi(\cdot; J_K, \tau)). \tag{4.5}$$

Fix a path p. Using local equivalence, we have either: (i) $\lambda(p; \sigma) = \lambda(p; \tau) = 0$, or (ii) $\lambda(p; \sigma) > 0$ and $\lambda(p; \tau) > 0$. In case (ii), $\lambda(p; \sigma) = \lambda(p; \tau)$, by (4.4), (4.5), and local equivalence again. Thus (i) and (ii) together establish that (4.2) and (4.3) are equal, as required. Q.E.D.

Corollary 4.4. Suppose Nature has perfect recall and all chance nodes are non-trivial. Fix probability measures σ and τ on Ω that are locally equivalent. Then a strategy s is globally optimal under σ if and only if it is globally optimal under τ.

Continuation of Example 4.2. Theorem 4.3 and Corollary 4.4 fail without perfect recall (for Nature). The measures σ and τ as above are locally equivalent. Yet T yields an expected payoff of 0 under σ, and an expected payoff of 3/2 under τ. (So B is globally optimal under σ, while T is globally optimal under τ.)

5 Discussion

Corollary 4.4 and Theorem 3.6 can be put together as follows: Assume perfect recall and non-triviality for both the decision maker and Nature. Then, to determine if a strategy is locally optimal, it is enough to know the localized probabilities of the decision maker.

For (even local) optimality, then, the analyst need only know how the decision maker sees the tree locally. We don't need to know the decision maker's global assessment of the tree.

But this does assume perfect recall and non-triviality. Perfect recall for the decision maker has a clear interpretation as a memory requirement (refer back to the end of Section 1 and also the references there). But what does perfect recall for Nature mean?[1] We'll give an answer for the game context, as analyzed under the epistemic approach.

For the application of the decision tree set-up (Definition 1.1) to a game, the decision maker is to be thought of as one player, Ann say. All the remaining players—Bob, Charlie, ...—are grouped together as Nature. This is because, under the epistemic approach, the strategies chosen by Bob, Charlie, ... are jointly uncertain as far as Ann is concerned, and so subject to joint probability assessment.

When, then, might Nature have perfect recall? One case is if there is just one other player, Bob, and he has perfect recall. The situation is different if there are two or more other players, even if each of these players has perfect recall. For example, suppose in Figure 5 that Ann chooses T or B, Bob chooses U or D, and Charlie chooses u or d. Then Bob and Charlie each has perfect recall, but if Ann assigns probability 1/2 to (U, u) and probability 1/2 to (D, d), there is no equivalent local assessment.

We could require Ann's global assessment to be a product of a global assessment of Bob's strategy and a global assessment of Charlie's strategy. Then, working with each assessment separately, we could find an equivalent local assessment. But an independence requirement like this is not in the spirit of the epistemic approach to games, which treats correlations as the norm.

[1] I am grateful to a referee for asking this question.

Of course, there will be special cases where 'overall' perfect recall still holds. (An obvious one is if the game has perfect information.) But, in general, we should work with global not local assessments by the players.

Appendix

Lemma A1. Suppose the decision maker has perfect recall and all decision nodes are non-trivial. Fix an information set I, nodes n_1 and n_2 in I, and an information set I' containing a node n_3 on the path from the root to n_1. Then there is a unique node in I' (not necessarily distinct from n_3) lying on the path from the root to n_2.

Proof. First note that by part (d.ii) of Definition 1.1, there cannot be more than one node in I' lying on the path from the root to n_2.

Now suppose, contra hypothesis, there is no node in I' lying on the path from the root to n_2. Let c denote the choice at n_3 that lies on the path to n_1. By non-triviality, there is a choice d, different from c, at I'. Construct a strategy s as follows: (i) at I', let s specify the choice d; (ii) at an information set crossed by the path from the root to n_2, let s specify the choice that lies on this path; (iii) at any other information set, let s be arbitrary. (Note that, by hypothesis, the information set I' does not fall under (ii), so s is well defined.) By construction, the node n_2 is allowed under s, while n_1 is not allowed under s. This contradicts perfect recall.

Q.E.D.

Lemma A2. Suppose the decision maker has perfect recall. Fix an information set I and nodes n_1 and n_2 in I. Fix also an information set I' containing nodes n_3 and n_4 (not necessarily distinct) where n_3 lies on the path from the root to n_1, and n_4 lies on the path from the root to n_2. Then the choice at n_3 that lies on the path to n_1 is the same as the choice at n_4 that lies on the path to n_2.

Proof. Let c be the choice at n_3 that lies on the path to n_1, and let d be the choice at n_4 that lies on the path to n_2. Suppose, contra hypothesis, that $c \neq d$. Construct a strategy s as follows: (i) at an information set crossed by the path from the root to n_2, let s specify the choice that lies on this path; (ii) at any other information set, let s be arbitrary. Note that s specifies d at I'. It follows that n_2 is allowed under s, while n_1 is not allowed under s. This contradicts perfect recall. Q.E.D.

We use Lemmas A1 and A2 in the proofs below of Lemmas 1.13 and 1.14 in the text. We also note that Lemma A1, together with part (d.ii) of Definition 1.1, easily implies the facts stated in Section 1: The precedence relation on information sets is irreflexive and transitive, and each information set I has at most one immediate predecessor.

Proof of Lemma 1.13. Since I' is allowed under s, there is a node n'_1 in I' and a state ω_1 such that the path induced by (s, ω_1) passes through n'_1. Likewise, since I' is allowed under s', there is a node n'_2 in I' and a state ω_2 such that the path induced by (s', ω_2) passes through n'_2. Since I precedes I', there are nodes n in I and n' in I' such that the path from the root to n' passes through n.

Lemma A1 then implies that there is a node n_1 in I (not necessarily distinct from n) lying on the path from the root to n'_1. That is, the path induced by (s, ω_1) passes through n_1. This establishes that I is allowed under s.

Likewise, Lemma A1 implies that there is a node n_2 in I (not necessarily distinct from n) lying on the path from the root to n'_2. That is, the path induced by (s', ω_2) passes through n_2. This establishes that I is allowed under s'.

Lemma A2 implies that the choice at n_1 that lies on the path to n'_1 is the same as the choice at n_2 that lies on the path to n'_2. Thus s and s' coincide at I. Q.E.D.

Proof of Lemma 1.14. Consider a state ω in $[I']$. By definition, there is a node n' in I' and a strategy s such that the path induced by (s, ω) passes through n'.

Since I precedes I', there are nodes n_1 in I and n_2 in I' such that the path from the root to n_2 passes through n_1.

Lemma A1 then implies there is a node n_3 in I (not necessarily distinct from n_1) such that the path from the root to n' passes through n_3. That is, the path induced by (s, ω) passes through n_3. Thus ω lies in $[I]$, as required. Q.E.D.

References

[AuHa$_4$92] R.J. Aumann & S. Hart, eds. *Handbook of Game Theory, Vol. 1.* North-Holland, 1992.

[Bo$_0$04] G. Bonanno. Memory and perfect recall in extensive games. *Games and Economic Behavior* 47(2):237–256, 2004.

[DrTu$_0$Wo$_0$57] M. Drescher, A. Tucker & P. Wolfe, eds. *Contributions to the Theory of Games, Vol. III*, *Annals of Mathematics Study* 39. Princeton University Press, 1957.

[Ha$_0$vMVa04] J. Halpern, R. van der Meyden & M. Vardi. Complete axiomatizations for reasoning about knowledge and time. *SIAM Journal on Computing* 33(3):674–703, 2004.

[$\mathrm{Ha_4}$92] S. Hart. Games in extensive and strategic form. In [$\mathrm{AuHa_4}$92, pp. 19–40].

[Is57] J. Isbell. Finitary games. In [$\mathrm{DrTu_0Wo_0}$57, pp. 79–96].

[$\mathrm{Kl_1}$02] J.J. Kline. Minimum memory for equivalence between *ex ante* optimality and time-consistency. *Games and Economic Behavior* 38(2):278–305, 2002.

[$\mathrm{Ku_0}$50] H. Kuhn. Extensive games. *Proceedings of the National Academy of Sciences* 36(10):570–576, 1950.

[$\mathrm{Ku_0}$53] H. Kuhn. Extensive games and the problem of information. In [$\mathrm{Ku_0Tu_0}$53, pp. 193–216].

[$\mathrm{Ku_0Tu_0}$53] H. Kuhn & A. Tucker, eds. *Contributions to the Theory of Games, Vol. II*, *Annals of Mathematics Study* 28. Princeton University Press, 1953.

[$\mathrm{Pi_0Ru_1}$97] M. Piccione & A. Rubinstein. On the interpretation of decision problems with imperfect recall. *Games and Economic Behavior* 20(1):3–24, 1997.

[$\mathrm{Ri_0}$99] K. Ritzberger. Recall in extensive games. *International Journal of Game Theory* 28(1):69–87, 1999.

[vB01a] J. van Benthem. Games in dynamic-epistemic logic. *Bulletin of Economic Research* 53(4):219–248, 2001.

[$\mathrm{Wi_0}$72] R. Wilson. Computing equilibria of two-person games from the extensive form. *Management Science* 18(7):448–460, 1972.

What Kind of Memory is Needed to Win Infinitary Muller Games?*

Erich Grädel
Łukasz Kaiser

Mathematische Grundlagen der Informatik
RWTH Aachen University
52056 Aachen, Germany
{graedel,kaiser}@informatik.rwth-aachen.de

Abstract

In an influential paper entitled *"How much memory is needed to win infinite games"*, Dziembowski, Jurdziński, and Walukiewicz have shown that there are Muller games of size $O(n)$ whose winning strategies require memory of size at least $n!$. This shows that the LAR-memory, based on the latest appearance records introduced by Gurevich and Harrington, is optimal for solving Muller games. We review these results and reexamine the situation for the case of infinitary Muller games, i.e. Muller games with infinitely many priorities. We introduce a new, infinite, memory structure, based on finite appearance records (FAR) and investigate classes of Muller games that can be solved with FAR-memory.

1 Introduction

We study two-player games of infinite duration that are played on finite or infinite game graphs. Such a game is *determined* if, from each position, one of the two players has a winning strategy. On the basis of the axiom of choice it is not difficult to prove that there exist nondetermined games. The classical theory of infinite games in descriptive set theory links determinacy of games with topological properties of the winning conditions. Usually the format of Gale-Stewart games is used where the two players strictly alternate, and in each move a player selects an element of $\{0, 1\}$; thus the outcome of a play is an infinite string $\pi \in \{0, 1\}^\omega$. Gale-Stewart games can be viewed as graph games, for instance on the infinite binary tree, or on a bipartite graph with four nodes. Zermelo [Ze13] proved already in 1913 that if in each play of a game, the winner is determined already after a finite number of moves, then one of the two players has a winning strategy.

* This research has been partially supported by the European Community Research Training Network "Games and Automata for Synthesis and Validation" (GAMES).

In topological terms the winning sets in such a game are clopen (open and closed). By a celebrated theorem due to Martin [$\mathrm{Ma}_7$75] every game where the winning condition is given by a Borel set is determined.

For game theory that relates to computer science, determinacy is just a first step in the analysis of a game. Rather than in the mere existence of winning strategies, one is interested in effective constructions of reasonably simple winning strategies. An aspect of crucial importance for the complexity of a strategy is its dependency on the history of the play.

In general, strategies may be very complicated functions that can depend on the entire history of the play. However, in many cases, simple strategies suffice. Of particular interest are *positional strategies* for which the next move depends only the current position, and not at all on previous history. That is, a player moving according to a positional strategy f will at a position v always perform the same move $v \to f(v)$ no matter how often and by what path position v has been reached. A game is *positionally determined*, if from each position, one of the two players has a positional winning strategy. Another important case are *finite-memory strategies* for which the dependency on the history can be calculated on the basis of a finite set of memory states and which can thus be implemented by a finite automaton.

Positional determinacy and determinacy via finite-memory strategies have been extensively studied for games whose winning conditions are defined in terms of a mapping that assigns to each position a *priority* from a finite set C. Specifically, in *Muller games* the winner of a play is determined by the set of those priorities that have been seen infinitely often. It has been proved by Gurevich and Harrington [$\mathrm{Gu}_1\mathrm{Ha}_2$82] that Muller games are determined via finite memory strategies that are based on a data structure called *latest appearance records* (LAR). Intuitively a latest appearance record is a list of priorities in the order in which they have last occurred in the play. Thus, on n priorities, an LAR-memory has $n!$ memory states. Dziembowski, Jurdziński, and Walukiewicz [$\mathrm{DzJu}_0\mathrm{Wa}_5$97] have shown that LAR-strategies are essentially optimal for Muller games.

Theorem 1.1. There exists a sequence $(\mathcal{G}_n)_{n\in\omega}$ of Muller games such that the game graph of $\mathcal{G}_n$ is of size $O(n)$ and every winning strategy for $\mathcal{G}_n$ requires a memory of size at least $n!$

In particular, Muller games need not be positionally determined, not even for solitaire games (where only one player moves). An important special case of Muller games are *parity games*. These are games with a priority labelling Ω assigning to each position v a priority $\Omega(v) \in \{0, \dots, d\}$, for some $d \in \mathbb{N}$, and with parity winning condition: Player 0 wins a play π if the least priority occurring infinitely often in π is even. Parity games are of importance for several reasons.

(1) Many classes of games arising in practical applications admit reductions to parity games (over larger game graphs). This is the case for games modelling reactive systems, with winning conditions specified in some temporal logic or in monadic second-order logic over infinite paths (S1S), for Muller games, but also for games with partial information appearing in the synthesis of distributed controllers.

(2) Parity games arise as the model checking games for fixed point logics such as the modal μ-calculus or LFP, the extension of first-order logic by least and greatest fixed points [$\mathrm{EmJu_1Si_3}$01, $\mathrm{Gr_0}$05]. In particular the model checking problem for the modal μ-calculus can be solved in polynomial time if, and only if, winning regions for parity games can be decided in polynomial time.

(3) Parity games are positionally determined [$\mathrm{EmJu_1}$91, $\mathrm{Mo_4}$91]. This is a game theoretical result of fundamental importance and with great algorithmic relevance.

To establish positional determinacy or finite-memory determinacy is a fundamental step in the analysis of an infinite game, and is also crucial for the algorithmic construction of winning strategies. In the case of parity games with finitely many priorities the positional determinacy immediately implies that winning regions can be decided in NP $\cap$ Co-NP; with a little more effort it follows that the problem is in fact in UP $\cap$ Co-UP [$\mathrm{Ju_0}$98]. Further, although it is not known yet whether parity games can be solved in polynomial time, all known approaches towards an efficient algorithmic solution make use of positional determinacy. The same is true for the efficient algorithms that we have for specific classes of parity games, including parity games with a bounded number of priorities [$\mathrm{Ju_0}$00], games where even and odd cycles do not intersect, solitaire games and nested solitaire games [$\mathrm{Be_2Gr_0}$04], and parity games of bounded tree width [Ob03], bounded entanglement [$\mathrm{Be_2Gr_0}$05], or bounded DAG-width [$\mathrm{Be_2}$+06, Ob06].

For several reasons it is interesting to generalise the theory of infinite games to the case of infinitely many priorities. Besides the theoretical interest, winning conditions depending on infinitely many priorities arise naturally in several contexts. In pushdown games, stack height and stack contents are natural parameters that may take infinitely many values. In [$\mathrm{Ca_0Du_0}$Th02], Cachat, Duparc, and Thomas study pushdown games with an infinity condition on stack contents, and Bouquet, Serre, and Walukiewicz [$\mathrm{Bo_4Se_1Wa_5}$03] consider more general winning conditions for pushdown games, combining a parity condition on the states of the underlying pushdown automaton with an unboundedness condition on stack heights. Similarly, Gimbert [$\mathrm{Gi_0}$04] considers games of bounded degree where the parity winning condition is combined with the requirement that an infinite portion of the game graph is visited.

A systematic study of positional determinacy of games with infinitely many priorities has been initiated in [$\text{Gr}_0\text{Wa}_5$06]. It has been shown that there are interesting cases where positional determinacy is a consequence of the winning condition only, holding for all game graphs. Most notably this is the case for the parity condition on ω. Moreover a complete classification of the infinitary Muller conditions with this property has been established in [$\text{Gr}_0\text{Wa}_5$06] and it has been shown that all of them are equivalent to a parity condition.

Whereas the proof for the positional determinacy of parity games with priorities in ω is somewhat involved, it is quite easy to construct games with infinitary Muller winning conditions whose winning strategies require infinite memory. For instance there are very simple max-parity games (where the maximal priority seen infinitely often determines the winner) with this property (see Section 4). Nevertheless, the required (infinite) memory structures are often quite simple. In some cases it is enough to store just the maximal priority seen so far. In other cases a tuple (of fixed length) of previously seen priorities suffices to determine the next move of a winning strategy. This motivates the introduction of a new memory structure for winning strategies, that we call *finite appearance records* (FAR) which generalise the LARs used for finitary Muller games. We determine some classes of Muller games that can be reduced to parity games via FAR-memories. These include games where the winning condition is a downward cone, a singleton condition, a finite union of upwards cones, or consists of finitely many winning sets only. Further the same property holds for all max-parity games where the difference between the priorities of any two consecutive positions is bounded.

Here is an outline of this paper. In Section 2 we present the technical definitions on games, winning strategies, memory structures and game reductions. In Section 3 we survey the case of Muller games with finitely many priorities and present proofs of two classical results of the field. First we show that Streett-Rabin games are positionally determined for one player (which also implies that parity games are positionally determined for both players). Second, we describe the LAR-memory and show how Muller games can be reduced, via LAR-memory, to parity games. In Section 4 we briefly survey the results from [$\text{Gr}_0\text{Wa}_5$06] on parity games and Muller games with infinitely many priorities. In Section 5 we introduce finite appearance records and FAR-memory structures. Finally, in Section 6 we analyse some classes of Muller games that can be solved with FAR-memories.

2 Games, strategies, and memory structures

We study infinite two-player games with complete information, specified by a triple $\mathcal{G} = (G, \Omega, W)$ where $G = (V, V_0, V_1, E)$ is a game graph, equipped with a partioning $V = V_0 \cup V_1$ of the nodes into positions of Player 0 and positions of Player 1, where $\Omega : V \to C$ is a function that assigns to each position a *priority* (or colour) from a set C, and where W specifies a *winning condition.* The pair $(\mathcal{G}, \Omega)$ is called the *arena* of the game. In case $(v, w) \in E$ we call w a successor of v and we denote the set of all successors of v by vE. To avoid tedious case distinctions, we assume that every position has at least one successor. A *play* in $\mathcal{G}$ is an infinite path $v_0v_1\dots$ formed by the two players starting from a given initial position v_0. Whenever the current position v_i belongs to V_0, then Player 0 chooses a successor $v_{i+1} \in v_iE$, if $v_i \in V_1$, then $v_{i+1} \in v_iE$ is selected by Player 1. The *winning condition* describes which of the infinite plays $v_0v_1\dots$ are won by Player 0, in terms of the sequence $\Omega(v_0)\Omega(v_1)\dots$ of priorities appearing in the play. Thus, a winning condition is given by a set $W \subseteq C^\omega$ of infinite sequences of priorities.

In traditional studies of infinite games it is usually assumed that the set C of priorities is finite, although the game graph itself (i.e., the set of positions) may well be infinite. This permits, for instance, to specify winning condition by formulae from a logic on infinite paths, such as LTL (linear time temporal logic), FO (first-order logic), or MSO (monadic second-order logic) over a vocabulary that uses the linear order $<$ and monadic predicates P_c for each priority $c \in C$.

A *(deterministic) strategy* for Player σ in a game $\mathcal{G} = (V, V_0, V_1, E, \Omega)$ is a partial function $f : V^*V_\sigma \to V$ that maps initial segments $v_0v_1\dots v_m$ of plays ending in a position $v_m \in V_\sigma$ to a successor $f(v_0\dots v_m) \in v_mE$. A play $v_0v_1\dots \in V^\omega$ is *consistent with* f, if Player σ always moves according to f, i.e., if $v_{m+1} = f(v_0\dots v_m)$ for every m with $v_m \in V_\sigma$. We say that such a strategy f is winning from position v_0, if every play that starts at v_0 and that is consistent with f, is won by Player σ. The *winning region* of Player σ, denoted W_σ, is the set of positions from which Player σ has a winning strategy.

A game $\mathcal{G}$ is *determined* if $W_0 \cup W_1 = V$, i.e., if from each position one of the two players has a winning strategy. In general, winning strategies can be very complicated. It is of interest to determine which games admit simple strategies, in particular *finite memory strategies* and *positional strategies.* While positional strategies only depend on the current position, not on the history of the play, finite memory strategies have access to bounded amount of information on the past. Finite memory strategies can be defined as strategies that are realisable by finite automata. However, we shall also need to consider strategies that require infinite memory. We therefore introduce a general notion of a memory structure and of a strategy with memory, generalising the finite memory strategies studied for instance in [DzJu$_0$Wa$_5$97].

Definition 2.1. A *memory structure* for a game $\mathcal{G}$ with positions in V is a triple $\mathfrak{M} = (M, \mathrm{update}, \mathrm{init})$, where M is a set of *memory states*, update : $M \times V \to M$ is a *memory update function* and init : $V \to M$ is a *memory initialisation function*. The *size* of the memory is the cardinality of the set M. A *strategy with memory* $\mathfrak{M}$ for Player σ is given by a next-move function $F : V_\sigma \times M \to V$ such that $F(v, m) \in vE$ for all $v \in V_\sigma, m \in M$. If a play, from starting position v_0, has gone through positions $v_0 v_1 \dots v_n$ the memory state is $m(v_0 \dots v_n)$, defined inductively by $m(v_0) = \mathrm{init}(v_0)$, and $m(v_0 \dots v_i v_{i+1}) = \mathrm{update}(m(v_0 \dots v_i), v_{i+1})$. In case $v_n \in V_\sigma$, the next move from $v_1 \dots v_n$, according to the strategy, leads to $F(v_n, m(v_0 \dots, v_n))$. In case $|M| = 1$, the strategy is positional; it can be described by a function $F : V_\sigma \to V$.

We will say that a game is determined via memory $\mathfrak{M}$ if it is determined and both players have winning strategies with memory $\mathfrak{M}$ on their winning regions. A game is *positionally determined* if it is determined via positional winning strategies.

Given a game graph $G = (V, V_0, V_1, E)$ and a memory structure $\mathfrak{M} = (M, \mathrm{update}, \mathrm{init})$ we obtain a new game graph $G \times \mathfrak{M} = (V \times M, V_0 \times M, V_1 \times M, E_{\mathrm{update}})$ where

$$E_{\mathrm{update}} = \{(v, m)(v', m') : (v, v') \in E \text{ and } m' = \mathrm{update}(m, v')\}.$$

Obviously, every play $(v_0, m_0)(v_1, m_1) \dots$ in $G \times \mathfrak{M}$ has a unique projection to the play $v_0 v_1 \dots$ in G. Conversely, every play $v_0, v_1, \dots$ in G has a unique extension to a play $(v_0, m_0)(v_1, m_1) \dots$ in $G \times \mathfrak{M}$ with $m_0 = \mathrm{init}(v_0)$ and $m_{i+1} = \mathrm{update}(m_i, v_{i+1})$.

Consider two games $\mathcal{G} = (G, \Omega, W)$ and $\mathcal{G}' = (G', \Omega', W')$. We say that $\mathcal{G}$ *reduces via memory* $\mathfrak{M}$ *to* $\mathcal{G}'$, (in short $\mathcal{G} \leq_{\mathfrak{M}} \mathcal{G}'$) if $G' = G \times \mathfrak{M}$ and every play in $\mathcal{G}'$ is won by the same player as the projected play in $\mathcal{G}$.

Given a memory structure $\mathfrak{M}$ for G and a memory structure $\mathfrak{M}'$ for $G \times \mathfrak{M}$ we obtain a memory structure $\mathfrak{M}^* = \mathfrak{M} \times \mathfrak{M}'$ for G. The set of memory locations is $M \times M'$ and we have memory initialization $\mathrm{init}^*(v) = (\mathrm{init}(v), \mathrm{init}'(v, \mathrm{init}(v))$ and the update function

$$\mathrm{update}^*((m, m'), v) := (\mathrm{update}(m, v), \mathrm{update}'(m', (v, \mathrm{update}(m, v))).$$

Proposition 2.2. Suppose that a game $\mathcal{G}$ reduces to $\mathcal{G}'$ via memory $\mathfrak{M}$ and that Player σ has a winning strategy for $\mathcal{G}'$ with memory $\mathfrak{M}'$ from $(v_0, \mathrm{init}(v_0)))$. Then Player σ has a winning strategy for $\mathcal{G}$ with memory $\mathfrak{M} \times \mathfrak{M}'$ from position v_0.

Proof. Given a strategy $F' : (V_\sigma \times M) \times M' \to (V \times M)$ for Player σ on $\mathcal{G}'$ we have to construct a strategy $F : (V_\sigma \times (M \times M')) \to V \times (M \times M')$.

For any pair $(v, m) \in V_\sigma \times M$ we have that $F'(v, m) = (w, \text{update}(m, w))$ where $w \in vE$. We now put $F(v, mm') = w$. If a play in $\mathcal{G}$ that is consistent with F proceeds from position v, with current memory location (m, m'), to a new position w, then the memory is updated to (n, n') with $n = \text{update}(m, w)$ and $n' = \text{update}'(m', (w, n))$. In the extended play in $\mathcal{G}'$ we have an associated move from position (v, m) to (w, n) with memory update from m' to n'. Thus, every play in $\mathcal{G}$ from initial position v_0 that is consistent with F is the projection of a play in $\mathcal{G}'$ from $(v_0, \text{init}(v_0))$ that is consistent with F'. Therefore, if F' is a winning strategy from $(v_0, \text{init}(v_0))$, then F is a winning strategy from v_0. Q.E.D.

Corollary 2.3. Every game that reduces via memory $\mathfrak{M}$ to a positionally determined game, is determined via memory $\mathfrak{M}$.

Obviously, memory reductions between games compose. If $\mathcal{G}$ reduces to $\mathcal{G}'$ with memory $\mathfrak{M} = (M, \text{update}, \text{init})$ and $\mathcal{G}'$ reduces to $\mathcal{G}''$ with memory $\mathfrak{M}' = (M', \text{init}', \text{update}')$ then $\mathcal{G}$ reduces to $\mathcal{G}''$ with memory $(M \times M', \text{init}'', \text{update}'')$ with $\text{init}''(v) = (\text{init}(v), \text{init}'(v, \text{init}(v)))$ and

$$\text{update}((m, m'), v) = (\text{update}(m, v), \text{update}'(m', (v, \text{update}(m, v)))).$$

3 Games with finitely many priorities

In this section we consider Muller games, Streett-Rabin games, and parity games with finitely many priorities.

3.1 Muller games and Streett-Rabin games

Definition 3.1. A *Muller winning condition* over a finite set C of priorities is written in the form $(\mathcal{F}_0, \mathcal{F}_1)$ where $\mathcal{F}_0 \subseteq \mathcal{P}(C)$ and $\mathcal{F}_1 = \mathcal{P}(C) \setminus \mathcal{F}_0$. A play π in a game with Muller winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ is won by Player σ if, and only if, $\text{Inf}(\pi)$, the set of priorities occurring infinitely in π, belongs to $\mathcal{F}_\sigma$.

The *Zielonka tree* for a Muller condition $(\mathcal{F}_0, \mathcal{F}_1)$ over C is a tree $Z(\mathcal{F}_0, \mathcal{F}_1)$ whose nodes are labelled with pairs (X, σ) such that $X \in \mathcal{F}_\sigma$. We define $Z(\mathcal{F}_0, \mathcal{F}_1)$ inductively as follows. Let $C \in \mathcal{F}_\sigma$ and $C_0, \ldots, C_{k-1}$ be the maximal sets in $\{X \subseteq C : X \in \mathcal{F}_{1-\sigma}\}$. Then $Z(\mathcal{F}_0, \mathcal{F}_1)$ consists of a root, labeled by (C, σ), to which we attach as subtrees the Zielonka trees $Z(\mathcal{F}_0 \cap \mathcal{P}(C_i), \mathcal{F}_1 \cap \mathcal{P}(C_i))$, for $i = 0, \ldots, k-1$.

Besides parity games there are other important special cases of Muller games. Of special relevance are games with Rabin and Streett conditions because these are positionally determined for one player [Kl$_0$94].

Definition 3.2. A *Streett-Rabin condition* is a Muller condition $(\mathcal{F}_0, \mathcal{F}_1)$ such that $\mathcal{F}_0$ is closed under union.

In the Zielonka tree for a Streett-Rabin condition, the nodes labeled $(X, 1)$ have only one successor. We remark that in the literature, Streett and Rabin conditions are often defined in a different manner, based on a collection $\{(E_i, F_i) : i = 1, \dots k\}$ of pairs of sets. However, it is not difficult to see that the definitions are equivalent [Zi98]. Further, it is also easy to show that if both $\mathcal{F}_0$ and $\mathcal{F}_1$ are closed under union, then $(\mathcal{F}_0, \mathcal{F}_1)$ is equivalent to a parity condition. The Zielonka tree for a parity condition is just a finite path.

In a Streett-Rabin game, Player 1 has a positional wining strategy on his winning region. On the other hand, Player 0 can win, on his winning region, via a finite memory strategy, and the size of the memory can be directly read of from the Zielonka tree. We present an elementary proof of this result. The exposition is inspired by [DzJu$_0$Wa$_5$97]. In the proof we use the notion of an attractor.

Definition 3.3. Let $\mathcal{G} = (V, V_0, V_1, E, \Omega)$ be an arena and let $X, Y \subseteq V$, such that X induces a subarena of $\mathcal{G}$ (i.e., every position in X has a successor in X). The *attractor* of Player σ of Y in X is the set $\mathrm{Attr}_\sigma^X(Y)$ of those positions $v \in X$ from which Player σ has a strategy in $\mathcal{G}$ to force the play into Y. More formally $\mathrm{Attr}_\sigma^X(Y) = \bigcup_\alpha Z^\alpha$ where

$$\begin{aligned} Z^0 &= X \cap Y, \\ Z^{\alpha+1} &= Z^\alpha \cup \{v \in V_\sigma \cap X : vE \cap Z^\alpha \neq \varnothing\} \cup \{v \in V_{1-\sigma} \cap X : vE \subseteq Z^\alpha\}, \\ Z^\lambda &= \bigcup_{\alpha<\lambda} Z^\alpha \quad \text{for limit ordinals } \lambda. \end{aligned}$$

On $\mathrm{Attr}_\sigma^X(Y)$, Player σ has a *positional attractor strategy* to bring the play into Y. Moreover $X \setminus \mathrm{Attr}_\sigma^X(Y)$ is again a subarena.

Theorem 3.4. Let $\mathcal{G} = (V, V_0, V_1, E, \Omega)$ be game with Streett-Rabin winning condition $(\mathcal{F}_0, \mathcal{F}_1)$. Then $\mathcal{G}$ is determined, i.e. $V = W_0 \cup W_1$, with a finite memory winning strategy of Player 0 on W_0, and a positional winning strategy of Player 1 on W_1. The size of the memory required by the winning strategy for Player 0 is bounded by the number of leaves of the Zielonka tree for $(\mathcal{F}_0, \mathcal{F}_1)$.

Proof. We proceed by induction on the number of priorities in C or, equivalently, the depth of the Zielonka tree $Z(\mathcal{F}_0, \mathcal{F}_1)$. Let ℓ be number of leaves of $Z(\mathcal{F}_0, \mathcal{F}_1)$. We distinguish two cases.

First, we assume that $C \in \mathcal{F}_1$. Let

$$X_0 := \{v : \text{Player 0 has a winning strategy} \\ \text{with memory of size} \leq \ell \text{ from } v\},$$

and $X_1 = V \setminus X_0$. It suffices to prove that Player 1 has a positional winning strategy on X_1. To construct this strategy, we combine three positional strategies of Player 1, a trap strategy, an attractor strategy, and a winning strategy on a subgame with fewer priorities.

We observe that X_1 is a trap for Player 0; this means that Player 1 has a positional trap-strategy t on X_1 to enforce that the play stays within X_1.

Since $\mathcal{F}_0$ is closed under union, there is a unique maximal subset $C' \subseteq C$ with $C' \in \mathcal{F}_0$. Let $Y := X_1 \cap \Omega^{-1}(C \setminus C')$ and let $Z = \mathrm{Attr}_1^{X_1}(Y) \setminus Y$. Observe that Player 1 has a positional attractor strategy a, by which he forces from any position $z \in Z$ that the play reaches Y.

Finally, let $V' = X_1 \setminus (Y \cup Z)$ and let $\mathcal{G}'$ be the subgame of $\mathcal{G}$ induced by V', with winning condition $(\mathcal{F}_0 \cap \mathcal{P}(C'), \mathcal{F}_1 \cap \mathcal{P}(C'))$. Since this game has fewer priorities, the induction hypothesis applies, i.e. $V' = W'_0 \cup W'_1$, Player 0 has a winning strategy with memory $\leq \ell$ on W'_0 and Player 1 has a positional winning strategy g' on W'_1. However, $W'_0 = \varnothing$; otherwise we could combine the strategies of Player 0 to obtain a winning strategy with memory $\leq \ell$ on $X_0 \cup W'_0 \supsetneq X_0$ contradicting the definition of X_0. Hence $W'_1 = V'$.

We can now define a positional strategy g for Player 1 on X_1 by

$$g(x) = \begin{cases} g'(x) & \text{if } x \in V', \\ a(x) & \text{if } x \in Z, \\ t(x) & \text{if } x \in Y. \end{cases}$$

Consider any play π that starts at a position $v \in X_1$ and is consistent with g. Obviously π stays within X_1. If it hits $Y \cup Z$ only finitely often, then from some point onward, it stays within V and coincides with a play consistent with g'. It is therefore won by Player 1. Otherwise π hits $Y \cup Z$, and hence also Y, infinitely often. Thus, $\mathrm{Inf}(\pi) \cap (C \setminus C') \neq \varnothing$ and therefore $\mathrm{Inf}(\pi) \in \mathcal{F}_1$.

We now consider the second case, $C \in \mathcal{F}_0$. There exist maximal subsets $C_0, \ldots, C_{k-1} \subseteq C$ with $C_i \in \mathcal{F}_1$. Observe that for every set $D \subseteq C$, we have that if $D \cap (C \setminus C_i) \neq \varnothing$ for all $i < k$, then $D \in \mathcal{F}_0$. Let

$$X_1 := \{v : \text{Player 1 has a positional winning strategy from } v\},$$

and $X_0 = V \setminus X_1$. We claim that Player 0 has a finite memory winning strategy of size $\leq \ell$ on X_0. To construct this strategy, we proceed in a similar way as above, for each of the sets $C \setminus C_i$. We will obtain strategies $f_0, \dots, f_{k-1}$ for Player 0, such that f_i has finite memory M_i, and we shall use these strategies to build a winning strategy f on X_0 with memory $M_0 \cup \dots \cup M_{k-1}$.

For $i = 0, \dots, k-1$, let $Y_i = X_0 \cap \Omega^{-1}(C \setminus C_i)$ let $Z_i = \mathrm{Attr}_0^{X_0}(Y_i) \setminus Y_i$, and let a_i be a positional attractor strategy, by which Player 0 can force a play from any position in Z_i to Y_i. Further, let $U_i = X_0 \setminus (Y_i \cup Z_i)$ and let $\mathcal{G}_i$ be the subgame of $\mathcal{G}$ induced by U_i, with winning condition $(\mathcal{F}_0 \cap \mathcal{P}(C_i), \mathcal{F}_1 \cap \mathcal{P}(C_i))$. The winning region of Player 1 in $\mathcal{G}_i$ is empty; indeed, if Player 1 could win $\mathcal{G}_i$ from v, then, by induction hypothesis, he could win with a positional winning strategy. By combining this strategy with the positional winning strategy of Player 1 on X_1, this would imply that $v \in X_1$; but $v \in U_i \subseteq V \setminus X_1$.

Hence, by induction hypothesis, Player 0 has a winning strategy f_i with finite memory M_i on U_i. Let $(f_i + a_i)$ be the combination of f_i with the attractor strategy a_i. From any position $v \in U_i \cup Z_i$ this strategy ensures that the play either remains inside U_i and is winning for Player 1, or it eventually reaches a position in Y_i.

We now combine the finite-memory strategies $(f_0+a_0), \dots, (f_{k-1}+a_{k-1})$ to a winning strategy f on X_0, which ensures that either the play ultimately remains within one of the regions U_i and coincides with a play according to f_i, or that it cycles infinitely often through all the regions $Y_0, \dots, Y_{k-1}$.

At positions in $\bigcap_{i<k} Y_i$, Player 0 just plays with a (positional) trap strategy ensuring that the play remains in X_0. At the first position $v \notin \bigcap_{i<k} Y_i$, Player 0 takes the minimal i such that $v \notin Y_i$, i.e. $v \in U_i \cup Z_i$, and uses the strategy $(f_i + a_i)$ until a position in $w \in Y_i$ is reached. At this point, Player 0 switches from i to $j = i + \ell \pmod k$ for the minimal ℓ such that $w \notin Y_j$. Hence $w \in U_j \cup Z_j$; Player 0 now plays with strategy $(f_j + a_j)$ until a position in Y_j is reached. There Player 0 again switches to the appropriate next strategy, and so on.

Assuming that $M_i \cap M_j = \varnothing$ for $i \neq j$ it is not difficult to see that f can be implemented with memory $M = M_0 \cup \dots \cup M_{k-1}$. We leave a formal definition of f to the reader.

It remains to prove that f is winning on X_0. Let π be a play that starts in X_0 and is consistent with f. If π eventually remains inside some U_i then it coincides, from some point onwards, with a play that is consistent with f_i, and therefore won by Player 0. Otherwise it hits each of the sets $Y_0, \dots, Y_{k-1}$ infinitely often. But this means that $\mathrm{Inf}(\pi) \cap (C \setminus C_i) \neq \varnothing$ for all $i \leq k$; as observed above this implies that $\mathrm{Inf}(\pi) \in \mathcal{F}_0$.

Note that, by the induction hypothesis, the size of the memory M_i is bounded by the number of leaves of the Zielonka subtrees $Z(\mathcal{F}_0 \cap \mathcal{P}(C_i), \mathcal{F}_1 \cap \mathcal{P}(C_i)$. Consequently the size of M is bounded by the number of leaves of $Z(\mathcal{F}_0, \mathcal{F}_1)$. Q.E.D.

Of course it also follows from this Theorem that parity games are positionally determined.

3.2 Latest appearance records and reductions for Muller games

The classical example of a game reduction with finite memory is the reduction of Muller games to parity games via latest appearance records. Intuitively, a latest appearance record (LAR) is a list of priorities ordered by their latest occurrence. More formally, for a finite set C of priorities, LAR(C) is the set of sequences $c_1 \dots c_k \natural c_{k+1} \dots c_\ell$ of elements from $C \cup \{\natural\}$ in which each priority $c \in C$ occurs at most once, and $\natural$ occurs precisely once. At a position v, the LAR $c_1 \dots c_k \natural c_{k+1} \dots c_\ell$ is updated by moving the priority $\Omega(v)$ to the end, and moving $\natural$ to the previous position of $\Omega(v)$ in the sequence. For instance, at a position with priority c_2, the LAR $c_1c_2c_3\natural c_4c_5$ is updated to $c_1\natural c_3c_4c_5c_2$. (If $\Omega(v)$ did not occur in the LAR, we simply append $\Omega(v)$ at the end.) Thus, the LAR-memory for an arena with priority labeling $\Omega : V \to C$ is the triple $(\mathrm{LAR}(C), \mathrm{init}, \mathrm{update})$ with $\mathrm{init}(v) = \natural\Omega(v)$ and

$$\mathrm{update}(c_1 \dots c_k \natural c_{k+1} \dots c_\ell, v) = c_1 \dots c_k \natural c_{k+1} \dots c_\ell \Omega(v)$$

in case $\Omega(v) \notin \{c_1 \dots c_\ell\}$, and

$$\mathrm{update}(c_1 \dots c_k \natural c_{k+1} \dots c_\ell, v) = c_1 \dots c_{m-1} \natural c_{m+1} \dots c_\ell c_m$$

if $\Omega(v) = c_m$.

The *hit-set* of a LAR $c_1 \dots c_k \natural c_{k+1} \dots c_\ell$ is the set $\{c_{k+1} \dots c_\ell\}$ of priorities occuring after the symbol $\natural$. Note that if in a play $\pi = v_0v_1 \dots$, the LAR at position v_n is $c_1 \dots c_k \natural c_{k+1} \dots c_\ell$ then $\Omega(v_n) = c_\ell$ and the hit-set $\{c_{k+1} \dots c_\ell\}$ is the set of priorities that have been seen since the latest previous occurrence of c_ℓ in the play.

Lemma 3.5. Let π be a play of a Muller game $\mathcal{G}$, and let $\mathrm{Inf}(\pi)$ be the set of priorities occurring infinitely often in π. On π the hit-set of the latest appearance record is, from some point onwards, always a subset of $\mathrm{Inf}(\pi)$ and infinitely often coincides with $\mathrm{Inf}(\pi)$.

Proof. For each play $\pi = v_0v_1v_2 \dots$ there is a position v_m such that $\Omega(v_n) \in \mathrm{Inf}(\pi)$ for all $n \geq m$. Since no priority outside $\mathrm{Inf}(\pi)$ is seen anymore after position v_m, the hit-set will from that point onwards always be contained in $\mathrm{Inf}(\pi)$, and the LAR will always have the form $c_1 \dots c_{j-1}c_j \dots c_k \natural c_{k+1} \dots c_\ell$ where $c_1, \dots c_{j-1}$ remain fixed and $\{c_j, \dots, c_k, c_{k+1}, \dots c_\ell\} = \mathrm{Inf}(\pi)$. Since all priorities in $\mathrm{Inf}(\pi)$ are seen again and again, it happens infinitely often that, among these, the one occuring leftmost in the LAR is hit. At such positions, the LAR is updated to $c_1, \dots, c_{j-1} \natural c_{j+1} \dots c_\ell c_j$ and the hit-set then coincides with $\mathrm{Inf}(\pi)$. Q.E.D.

Theorem 3.6. Every Muller game with finitely many priorities reduces via LAR memory to a parity game.

Proof. Let $\mathcal{G}$ be a Muller game with game graph G, priority labelling $\Omega : V \to C$ and winning condition $(\mathcal{F}_0, \mathcal{F}_1)$. We have to prove that $\mathcal{G} \leq_{\text{LAR}} \mathcal{G}'$ for a parity game $\mathcal{G}'$ with game graph $G \times \text{LAR}(C)$ and an appropriate priority labeling Ω' on $V \times \text{LAR}(C)$ which is defined as follows.

$$\Omega'(v, c_1 c_2 \dots c_k \natural c_{k+1} \dots c_\ell) = \begin{cases} 2k & \text{if } \{c_{k+1}, \dots, c_\ell\} \in \mathcal{F}_0, \\ 2k+1 & \text{if } \{c_{k+1}, \dots, c_\ell\} \in \mathcal{F}_1. \end{cases}$$

Let $\pi = v_0 v_1 v_2 \dots$ be a play on $\mathcal{G}$ and fix a number m such that, for all numbers $n \geq m$ and $\Omega(v_n) \in \text{Inf}(\pi)$, the LAR at position v_n has the form $c_1 \dots c_j c_{j+1} \dots c_k \natural c_{k+1} \dots c_\ell$ where $\text{Inf}(\pi) = \{c_{j+1}, \dots c_\ell\}$ and the prefix $c_1 \dots c_j$ remains fixed. In the extended play $\pi' = (v_0 r_0)(v_1, r_1) \dots$ all nodes (v_n, r_n) for $n \geq$ will therefore have a priority $2k + \rho$ with $k \geq j$ and $\rho \in \{0, 1\}$. Assume that the play π is won by Player σ, i.e., $\text{Inf}(\pi) \in \mathcal{F}_\sigma$. Since infinitely often the hit-set of the LAR coincides with $\text{Inf}(\pi)$, the minimal priority seen infinitely often on the extended play is $2j + \sigma$. Thus the extended play in the parity game $\mathcal{G}'$ is won by the same player as the original play in the Muller game $\mathcal{G}$. Q.E.D.

4 Games with infinitely many priorities

The definition of Muller games (Definition 3.1) directly generalises to countable sets C of priorities[1]. However, a representation of a Muller condition by a Zielonka tree is not always possible, since we may have sets $D \in \mathcal{F}_\sigma$ that have subsets in $\mathcal{F}_{1-\sigma}$ but no maximal ones. Further, it turns out that the condition that $\mathcal{F}_0$ and $\mathcal{F}_1$ are both closed under finite unions is no longer sufficient for positional determinacy. To see this let us discuss the possible generalisations of parity games to the case of priority assigments $\Omega : V \to \omega$. For parity games with finitely many priorities it is of course purely a matter of taste whether we let the winner be determined by the least priority seen infinitely often or by the greatest one. Here this is no longer the case. Based on priority assignments $\Omega : V \to \omega$ we consider the following classes of games.

Infinity games are games where Player 0 wins those infinite plays in which no priority at all appears infinitely often, i.e.

$$\mathcal{F}_0 = \{\varnothing\},$$
$$\mathcal{F}_1 = \mathcal{P}(\omega) \setminus \{\varnothing\}.$$

[1] With minor modifications, it can also be generalised to uncountable sets C. See [Gr$_0$Wa$_5$06] for a discussion of this.

Parity games are games where Player 0 wins the plays in which the least priority seen infinitely often is even, or where no priority appears infinitely often. Thus,

$$\mathcal{F}_0 = \{X \subseteq \omega : \min(X) \text{ is even}\} \cup \{\varnothing\},$$
$$\mathcal{F}_1 = \{X \subseteq \omega : \min(X) \text{ is odd}\}.$$

Max-parity games are games where Player 0 wins if the maximal priority occurring infinitely often is even, or does not exist, i.e.

$$\mathcal{F}_0 = \{X \subseteq \omega : \text{ if } X \text{ is finite and non-empty, then } \max(X) \text{ is even}\},$$
$$\mathcal{F}_1 = \{X \subseteq \omega : X \text{ is finite, non-empty, and } \max(X) \text{ is odd}\}.$$

It is easy to see that infinity games are a special case of parity games (via a simple reassignment of priorities). Further we note that for both parity games and max-parity games, $\mathcal{F}_0$ and $\mathcal{F}_1$ are closed under finite unions. Nevertheless the conditions behave quite differently. The parity condition has a very simple Zielonka tree, namely just a Zielonka path

$$\omega \longrightarrow \omega \setminus \{0\} \longrightarrow \omega \setminus \{0,1\} \longrightarrow \omega \setminus \{0,1,2\} \longrightarrow \cdots$$

whereas there is no Zielonka tree for the max-parity condition since $\omega \in \mathcal{F}_0$ has no maximal subset in $\mathcal{F}_1$ (and $\mathcal{F}_1$ is not closed under unions of chains). This is in fact related to a much more important difference concerning the memory needed for winning strategies.

Proposition 4.1. Max-parity games with infinitely many priorities in general do not admit finite memory winning strategies.

Proof. Consider the max-parity game with positions $V_0 = \{0\}$ and $V_1 = \{2n+1 : n \in \mathbb{N}\}$ (where the name of a position is also its priority), such that Player 0 can move from 0 to any position $2n+1$ and Player 1 can move back from $2n+1$ to 0. Clearly Player 0 has a winning strategy from each position but no winning stategy with finite memory. Q.E.D.

On the other hand it has been shown in [Gr$_0$Wa$_5$06] that infinity games and parity games with priorities in ω do admit positional winning strategies for both players on all game graphs. In fact, parity games over ω turn out to be the only Muller games with this property.

Theorem 4.2 (Grädel, Walukiewicz)**.** Let $(\mathcal{F}_0, \mathcal{F}_1)$ be a Muller winning condition over a countable set C of priorities. Then the following are equivalent.

(i) Every game with winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ is positionally determined.

(ii) Both $\mathcal{F}_0$ and $\mathcal{F}_1$ are closed under finite unions, unions of chains, and non-empty intersections of chains.

(iii) The Zielonka tree of $(\mathcal{F}_0, \mathcal{F}_1)$ exists, and is a path of co-finite sets (and possibly the empty set at the end).

(iv) $(\mathcal{F}_0, \mathcal{F}_1)$ reduces to a parity condition over $n \leq \omega$ priorities.

5 Finite Appearance Records

Although over an infinite set of priorities one can easily define Muller games that do not admit finite memory strategies, these games are often solvable by strategies with very simple infinite memory structures. For instance, for the max-parity game described in the proof of Proposition 4.1, it suffices for Player 0 to store the maximal priority seen so far, in order to determine the next move in her winning strategy. One can readily come up with other games where the memory required by a winning strategies is essentially a finite collection of previously seen priorities.

This motivates the definition of an infinite memory structure that we call *finite appearance records* (FAR) which generalises the LAR-memory for finitely coloured games. In a FAR we store tuples of previously encountered priorities or some other symbols from a finite set. Additionally the update function in the appearance record is restricted, so that new values of the memory can be equal only to the values stored before or to the currently seen priority.

Definition 5.1. A d-dimensional FAR-memory for a game $\mathcal{G}$ with priorities in C is a memory structure $(M, \text{update}, \text{init})$ for $\mathcal{G}$ with $M = (C \cup N)^d$ for some finite set N such that whenever

$$\text{update}(m_1, \ldots, m_d, v) = (m'_1, \ldots, m'_d)$$

then $m'_i \in \{m_1, \ldots, m_d\} \cup N \cup \{\Omega(v)\}$.

Note that an LAR-memory over a finite set C is a special case of an FAR-memory, with $d = |C| + 1$ and $N = \{\natural, B\}$, where B is a blank symbol used to pad latest appearance records in which some priorities are missing. Here the dimension of the FAR depends on the size of C. Hence, the question arises whether there is a fixed dimension d and a fixed additional set N such that every finitely coloured Muller game reduces to a parity game via d-dimensional FAR-memory. From Theorem 1.1 it follows that his is not the case. Indeed, since $n!$ grows faster than n^d for any constant d, we infer that for any dimension d there is a Muller game $\mathcal{G}_d$ that can not be reduced to a parity game via d-dimensional FAR-memory.

From this we obtain the following conclusion.

Proposition 5.2. There exists an infinitely coloured Muller game $\mathcal{G}$ that does not reduce to a parity game with any FAR-memory.

Proof. Take $\mathcal{G}$ to be the disjoint sum of the games $\mathcal{G}_d$, assuming that all these games have disjoint sets of priorities. Suppose that $\mathcal{G}$ reduces to a parity game via some FAR-memory of dimension d. Since game extensions preserve connectivity it follows that the extension of the connected component $\mathcal{G}_d$ of $\mathcal{G}$ will also be a parity game. But this contradicts the fact that $\mathcal{G}_d$ does not reduce to a parity game via d-dimensional FAR-memory. Q.E.D.

6 FAR-reductions for Muller games

In this section we consider some cases of Muller games with priorities in ω that admit FAR-reductions to positionally determined games.

To illustrate the idea consider any *downwards cone* $\mathcal{F}_0 = \{X : X \subseteq A\}$ for a fixed set $A \subseteq \omega$. Again it is easy to see that such games may require infinite-memory strategies. To reduce such a game to a parity game $\mathcal{G}'$ it suffices to store the maximal priority m seen so far, and to define priorities in $\mathcal{G}'$ by

$$\Omega'(v, m) = \begin{cases} 2m + 2 & \text{if } \Omega(v) \in A, \\ 2\Omega(v) + 1 & \text{otherwise.} \end{cases}$$

If $\mathrm{Inf}(\pi) \subseteq A$ then Player 0 wins π' since no odd priority is seen infinitely often in π'. If there is some $a \in \mathrm{Inf}(\pi) \setminus A$, then $2a+1$ occurs infinitely often in π', and since $a \leq m$ from some point onwards, no smaller even priority can have this property, so Player 1 wins π'.

Hence any Muller game such that $\mathcal{F}_0$ (or $\mathcal{F}_1$) is a downwards cone is determined via one-dimensional FAR-memory.

6.1 Visiting sequences and singleton Muller conditions

Our next example for winning conditions that are amenable for an approach via FAR-reductions are Muller games where the winning condition of Player 0 is a singleton, i.e., $\mathcal{F}_0 = \{A\}$, $\mathcal{F}_1 = \mathcal{P}(\omega) \setminus \{A\}$.

We first observe that such games may require infinite memory.

Theorem 6.1. For any $A \neq \varnothing$, there exists a (solitaire) Muller game with $\mathcal{F}_0 = \{A\}$ whose winning strategies all require infinite memory.

Proof. If $A = \{a_1, a_2, \dots\}$ is infinite, take the game with set of positions $V = V_0 = A$ (where the name of a position indicates also its priority), and moves (a_1, a_n) and (a_n, a_1) for all $n \geq 2$. If $A = \{a_1, \dots, a_n\}$ is finite, let

$\omega\setminus A = \{b_1, b_2, \dots\}$ we consider instead the game with $V = V_0 = A \cup (\omega \setminus A)$, and set of moves

$$E = \{(a_i, a_{i+1}) : 1 \leq i < n\} \cup \{(a_n, b) : b \in (\omega \setminus A)\} \cup \{(b, a_1) : b \in (\omega \setminus A)\}.$$

In both cases, Player 0 wins, but requires infinite memory to do so. Q.E.D.

We will prove that singleton Muller games can be reduced via FAR-memory to parity games with priorities in ω which, as shown in [Gr$_0$Wa$_5$06], are positionally determined. The FAR-memory that we use for this reduction is based on a particular order in which the elements of the winning sets have to be seen infinitely often, which is specified by a visiting sequence.

Definition 6.2. Let $A = \{a_1 < a_2 < \dots\}$ be an infinite subset of ω. For each $n \in \omega$, let $p(a_n) := a_1 a_2 \dots a_n$ be the *prefix* of a_n. The *visiting sequence* of A is the concatenation of the prefixes of all elements of A

$$\text{visit}(A) = p(a_1)p(a_2)p(a_3)\dots$$

For a finite set $\{a_1 < a_2 < \dots < a_n\} \subseteq \omega$ we define $\text{visit}(A) = p(a_n)^\omega$.

Let $\mathcal{G}$ be a Muller game over ω.

Lemma 6.3. For any play $\pi = v_1 v_2 \dots$ of $\mathcal{G}$ the set $\text{Inf}(\pi)$ is the unique set A with the following two properties:

(1) There is a sequence of indices $i_1 < i_2 < \dots$ such that $\Omega(v_{i_1})\Omega(v_{i_2})\dots$ forms the visiting sequence of A.

(2) If $\Omega(v_k) \in \omega \setminus A$ then there is only a finite number of indices $i > k$ such that $\Omega(v_i) \in \{0, \dots, \Omega(v_k)\} \cap \omega \setminus A$.

Proof. First we notice that $A = \text{Inf}(\pi)$ indeed fulfils these two properties. The visiting sequence can be chosen from the play as all elements of $\text{Inf}(\pi)$ appear infinitely often. Since all elements of $\omega \setminus \text{Inf}(\pi)$ occur only finitely often in the play, the second property must also hold.

Conversely, if a set A satisfies property (1), then all elements of A appear infinitely often in π, so $A \subseteq \text{Inf}(\pi)$. If there were an element $a \in \text{Inf}(\pi) \setminus A$, then for any k with $\Omega(v_k) = a$, there were infinitely many indices $i > k$, with $\Omega(v_i) = a$ which contradicts property (2). Thus if A satisfies properties (1) and (2), then $A = \text{Inf}(\pi)$. Q.E.D.

Let $A \subseteq \omega$ be infinite. Any initial segment of the visiting sequence of A can be written in the form $p(a_1)p(a_2)\dots p(a_i)a_1 a_2 \dots a_j$ where $1 \leq j \leq i+1$. It can be represented by a pair (p, c) where $c = a_j$ indicates the position of the last letter in the current prefix $p(a_{i+1})$, and $p = a_i$ indicates the last previously completed prefix (or ε if we are at the first element). For instance, the initial segment $a_1\, a_1a_2\, a_1a_2a_3\, a_1a_2a_3$ of the visiting sequence of A is encoded by (a_3, a_3), the initial segment a_1 is encoded by (ε, a_1), and the empty initial segment by $(\varepsilon, \varepsilon)$. We write $\mathrm{visit}_n(A)$ for the initial segment of length A of $\mathrm{visit}(A)$.

Given a (finite or infinite) winning set A, we want to use a three-dimensional FAR-memory to check whether $\mathrm{Inf}(\pi) = A$. For infinite A, the memory state after an initial segment of a play is a triple (p, c, q) where (p, c) encode the initial segment of the visiting sequence of A that has been seen so far, and q is the maximal priority that has occurred.

Definition 6.4. For any infinite set $A \subseteq \omega$, we define a three-dimensional FAR-memory $\mathrm{FAR}(A) = (M, \mathrm{init}, \mathrm{update})$ with $M = \{(p, c, q) : p, c \in \omega \cup \{\varepsilon\}, q \in \omega\}$. The initialisation function is defined by

$$\mathrm{init}(v) = \begin{cases} (\varepsilon, \Omega(v), \Omega(v)) & \text{if } \Omega(v) = a_1, \\ (\varepsilon, \varepsilon, \Omega(v)) & \text{if } \Omega(v) \neq a_1. \end{cases}.$$

The update function is defined by

$$\mathrm{update}(p, c, q, v) := (p', c', q'),$$

where $q' = \max(q, \Omega(v))$, and where either (p, c) and respectively (p', c') encode, for some n, the initial segments $\mathrm{visit}_n(A)$ and $\mathrm{visit}_{n+1}(A)$ of the visiting sequence of A such that $\mathrm{visit}_{n+1}(A) = \mathrm{visit}_n(A)\Omega(v)$, or otherwise, $(p', c') = (p, c)$.

For a more formal description, let

$$\mathrm{up}(p, c, v) = \begin{cases} 2 & \text{if, for some } i,\ p = a_i, c = a_{i+1}, \Omega(v) = a_1, \\ 1 & \text{if, for some } j \leq i,\ p = a_i, c = a_j, \Omega(v) = a_{j+1}, \\ 0 & \text{otherwise} \end{cases}$$

(where, to simplify notation, we identify ε with a_0). Note that $\mathrm{up}(p, c, v) = 2$ if, at node v, the visiting sequence is updated with an a_1 (i.e. a prefix $p(a_i)$ has been completed and a new one is started), that $\mathrm{up}(p, c, v) = 1$ if the visiting sequence is updated by another value, and that $\mathrm{up}(p, c, v) = 0$ if no update of the visiting sequence happens at v. Then we can define $\mathrm{update}(p, c, q, v) := (p', c', q')$ by

$$\begin{aligned} (p', c') &= \begin{cases} (c, \Omega(v)) & \text{if } \mathrm{up}(p, c, v) = 2, \\ (p, \Omega(v)) & \text{if } \mathrm{up}(p, c, v) = 1, \\ (p, c) & \text{if } \mathrm{up}(p, c, v) = 0, \end{cases} \\ q' &= \max(q, \Omega(v)). \end{aligned}$$

For finite $A = \{a_1 < a_2 < \cdots < a_n\}$ this has to be modified since once cannot really encode the part of the visiting sequence that one has seen with priorities in A. In this case the value (p, c, q) is so that c is the last element of the visiting sequence, q is the maximal priority that has occurred so far, and p is the maximal priority that had occured up to the last time when, in the visiting sequence of A, a prefix $p(a_n)$ had been completed and c had been updated from a_n to a_1. Thus we set

$$\mathrm{up}(p, c, v) = \begin{cases} 2 & \text{if } c = a_n, \Omega(v) = a_1, \\ 1 & \text{if, for some } i < n, c = a_i, \Omega(v) = a_{i+1}, \\ 0 & \text{otherwise,} \end{cases}$$

and $\mathrm{update}(p, c, q, v) := (p', c', q')$ with

$$\begin{aligned} (p', c') &= \begin{cases} (q, \Omega(v)) & \text{if } \mathrm{up}(p, c, v) = 2. \\ (p, \Omega(v)) & \text{if } \mathrm{up}(p, c, v) = 1, \\ (p, c) & \text{if } \mathrm{up}(p, c, v) = 0, \end{cases} \\ q' &= \max(q, \Omega(v)). \end{aligned}$$

Theorem 6.5. Any singleton Muller game with $\mathcal{F}_0 = \{A\}$ can be reduced, via memory $\mathrm{FAR}(A)$, to a parity game.

Proof. The given Muller game $\mathcal{G}$ with arena (G, Ω) and Muller condition such that $\mathcal{F}_0 = \{A\}$ is reduced via memory $\mathrm{FAR}(A)$ to a parity game $\mathcal{G}'$ with priority function $\Omega' : V \times \mathrm{FAR}(A) \to \omega$ defined as follows:

$$\Omega'(v, p, c, q) = \begin{cases} 2p + 2 & \text{if } \Omega(v) \in A, \mathrm{up}(p, c, v) \in \{1, 2\}, \\ 2p + 3 & \text{if } \Omega(v) \in A, \mathrm{up}(p, c, v) = 0, \\ \min(2p + 3, 2\Omega(v) + 1) & \text{if } \Omega(v) \notin A. \end{cases}$$

We have to prove that any play $\pi = v_0 v_1 v_2 \dots$ of $\mathcal{G}$ is won by the same player as the extended play

$$\pi' = (v_0, p_0, c_0, q_0)(v_1, p_1, c_1, q_1)(v_2, p_2, c_2, q_2) \dots$$

of $\mathcal{G}'$.

We first assume that $\mathrm{Inf}(\pi) = A$ and prove that either no priority at all occurs infinitely often in π' or the minimal such is even. If A is infinite, then the sequence of the values p_n diverges and therefore no priority will be seen infinitely often in π'. If A is finite then it may be the case that the sequence $(p_n)_{n \in \omega}$ converges, i.e., $p_n = p$ from some point onwards. But since the visiting sequence will be updated again and again this means that infinitely often the priority $2p+2$ occurs in π', and the only other priority that may occur infinitely often is $2p+3$. Hence Player 0 wins π'.

For the converse, we assume that Player 1 wins π. We distinguish several cases. If there exist some $a \in A \setminus \mathrm{Inf}(\pi)$ then from some point onwards, the visiting sequence cannot be updated anymore, so the sequence $(p_n)_{n\in\omega}$ stabilises at some value p. Then the minimal priority seen infinitely often is either $2p+3$, or $2\Omega(v)+1$ for some $\Omega(v) \in \omega \setminus A$ and Player 1 also wins π'. If no such element a exists, then $A \subsetneq \mathrm{Inf}(\pi)$ and there is a minimal element $b \in \mathrm{Inf}(\pi) \setminus A$. If the sequence $(p_n)_{n\in\omega}$ diverges (which is always the case for infinite winning sets A) then the minimal priority seen infinitely often in π' is $2b+1$. If A is finite then the sequence p_n may stabilise at some value p which coincides with the largest priority ever occurring in π. Hence $b \leq p$ and therefore $2b+1 < 2p+2$, so the minimal priority seen infinitely often in π' is $2b+1$. Again Player 1 wins the associated play in the parity game. Q.E.D.

Corollary 6.6. Singleton Muller games are determined with FAR memory.

6.2 Finite unions of upwards cones

Visiting sequences can also be used for the case where $\mathcal{F}_0$ is a finite union of upwards cones, i.e.

$$\mathcal{F}_0 = \bigcup_{i=1}^{k} \{X : A_i \subseteq X \subseteq \omega\}$$

for some finite collection of sets $A_1, \ldots, A_k$.

The FAR-memory stores the pairs (p_i, c_i) encoding the visiting sequences of $A_1, \ldots, A_k$. All that has to checked is whether $A_i \subseteq \mathrm{Inf}(\pi)$ for some i, which is the case if, and only if, one of the visiting sequences is updated infinitely often. Thus we can define priorities by

$$\Omega'(v, p_1, c_1, \ldots, p_k, c_k) = \begin{cases} 0 & \text{if } \mathrm{up}(p_i, c_i, v) = 2 \text{ for some } i, \\ 1 & \text{otherwise.} \end{cases}$$

Theorem 6.7. Any Muller game such that $\mathcal{F}_\sigma$ is a finite union of upwards cones is determined via FAR-memory.

6.3 Muller conditions with finitely many winning sets

We now consider the case of Muller games whose winning conditions are defined by a finite collection of (possibly infinite) sets, $\mathcal{F}_0 = \{A_1, \ldots, A_k\}$. To extend the idea presented above to this case we are going to use the memory $\mathrm{FAR}(A_i)$ for each set A_i and additionally we have to remember when the set A_i is *active*, as is described below. The property of being active is stored in a value $a_i \in \{0, 1, 2\}$.

Definition 6.8. For any finite collection $\{A_1, \ldots, A_k\}$ of sets $A_i \subseteq \omega$, we define a $4k$-dimensional FAR-memory $\mathrm{FAR}(A_1, \ldots, A_k) = (M, \mathrm{init}, \mathrm{update})$. We denote the FAR-memory of A_i by $\mathrm{FAR}(A_i) = (M_i, \mathrm{init}_i, \mathrm{update}_i)$. Then $M = M_1 \times M_2 \times \ldots \times M_k \times \{0,1,2\}^k$. The initialisation function is defined by

$$\mathrm{init}(v) = (\mathrm{init}_1(v), \ldots, \mathrm{init}_k(v), \overline{0}).$$

The update function is defined by

$$\begin{aligned}\mathrm{update}&(m_1, \ldots, m_k, a_1, \ldots, a_k, v) \\ &= (\mathrm{update}_1(m_1, v), \ldots, \mathrm{update}_k(m_k, v), a'_1, \ldots, a'_k),\end{aligned}$$

where a'_i is the new activation value for sequence i defined by

$$a'_i = \begin{cases} 0 & \text{if } v \notin A_i \text{ and for some } j \leq k \ \mathrm{up}_j(m_j, v) > 0, \\ \min(2, a_i + 1) & \text{if } \mathrm{up}_i(m_i, v) = 2, \\ a_i & \text{otherwise.} \end{cases}$$

Theorem 6.9. Any Muller game with $\mathcal{F}_0 = \{A_1, \ldots, A_k\}$ can be reduced, via memory $\mathrm{FAR}(A_1, \ldots, A_k)$, to a parity game.

Proof. The given Muller game $\mathcal{G}$ with arena (G, Ω) and Muller condition such that $\mathcal{F}_0 = \{A_1, \ldots, A_k\}$ is reduced to a parity game $\mathcal{G}'$ with priority function Ω' defined as follows:

$$\Omega'(v, \overline{m}, \overline{a}) = \begin{cases} \max_{\{i \,:\, \Omega(v) \in A_i \wedge a_i = 2\}}(2kp_i + 2r_i + 2) & \text{if } j \text{ exists such that } \Omega(v) \in A_j, a_j = 2, \mathrm{up}_j(m_j, v) \in \{1,2\}, \\ \max_{\{i \,:\, \Omega(v) \in A_i \wedge a_i = 2\}}(2kp_i + 2r_i + 3) & \text{if } j \text{ exists such that } \Omega(v) \in A_j, a_j = 2, \mathrm{up}_j(m_j, v) = 0 \text{ for all such } j, \\ \min(2k \max(p_1 \ldots p_k) + 3, 2\Omega(v) + 1) & \text{otherwise,} \end{cases}$$

where p_i is the first component of the i-th memory $m_i = (p_i, c_i, q_i)$ and for each $A_i \in \mathcal{F}_0$ we have $r_i = |\{A_j \in \mathcal{F}_0 : A_i \subseteq A_j\}|$.

We have to prove that any play $\pi = v_0 v_1 v_2 \ldots$ of $\mathcal{G}$ is won by the same player as the extended play

$$\pi' = (v_0, m_{10}, \ldots, m_{k0}, a_{10}, \ldots a_{k0})(v_1, m_{11}, \ldots, m_{k1}, a_{11}, \ldots a_{k1}) \ldots.$$

For a given play π of $\mathcal{G}$, we divide the sets $A_1, \ldots, A_k \in \mathcal{F}_0$ into three classes.

The good: A_i is a good set if A_i is active (i.e., $a_i = 2$) only finitely often in π.

The bad: A_i is a bad set, if $A_i \subseteq \text{Inf}(\pi)$ and A_i is not a good set.

The ugly: A_i is an ugly set if there is a priority $c \in A_i \setminus \text{Inf}(\pi)$ and A_i is not a good set.

Lemma 6.10. If A_i is bad and A_j is ugly, then $A_i \subseteq A_j$.

Proof. Assume that there is a $b \in A_i \setminus A_j$. Since $A_i \subseteq \text{Inf}(\pi)$ the visiting sequence for A_i is updated infinitely often, hence infinitely often with b, and whenever this happens then a_j is reset to 0. By definition there is a $c \in A_j$ that is seen only finitely many times in π. Therefore $a_j = 0$ from some point onwards. But this contradicts the assumption that A_j is not good. Q.E.D.

We first assume that $\text{Inf}(\pi) = A_i$ and prove that either no priority at all occurs infinitely often in π' or the minimal such priority is even.

Since from some point on there is no priority $d \notin A_i$ that occurs infinitely often, then for all sets A_j that are not subsets of A_i the visiting sequence will not be updated any more, and so the sequence $(p_{jn})_{n\in\omega}$ stabilises at some value p_j. Since the visiting sequence of A_i is updated infinitely often, we get that from some point on $a_i = 2$. Hence A_i is a bad set. We can now argue as in the proof of Theorem 6.5: if infinitely many priorities appear in π, then the sequence $(p_{in})_{n\in\omega}$ diverges and no priority at all will be seen infinitely often in π'. It remains to consider the case where only finitely many priorities occur in π. Then the sequence $(p_{in})_{n\in\omega}$ stabilises at some value p, which is the maximal priority appearing in π. For any $A_j \subsetneq A_i$, the sequence $(p_{jn})_{n\in\omega}$ will then also stabilise at the same value p, and $r_j > r_i$. It follows that some priority of form $2kp + 2r_\ell + 2$ occurs infinitely often in π', where $r_\ell \geq r_i$.

Suppose now that some smaller odd priority occurs infinitely often in π'. Then it would have to be of the form $2kp + 2r_j + 3$ with $r_j < r_\ell$ such that $a_j = 2$ infinitely often. However, only finitely many priorities appear in π. Hence if there are infinitely many positions v such that $\Omega(v) \in A_j$ and $a_j = 2$, then from some point onwards all these positions v satisfy that $\Omega(v) \in A_j \cap A_i$ and $a_i = 2$. On infinitely many such positions an update happens, and therefore, also the priority $2kp + 2r_j + 2$ appears infinitely often. Hence Player 0 wins π'.

For the converse, we now assume that Player 1 wins π.

Lemma 6.11. Suppose that some even priority $2kq+2r+2$ is seen infinitely often in π'. Then q is the maximal priority that occurs in π and $r = r_\ell$ for some bad set A_ℓ.

Proof. If there are infinitely many occurrences of $2kq + 2r + 2$ in π', then q is the maximal priority that occurs in π and some A_i is updated infinitely often (i.e. $A_i \subseteq \mathrm{Inf}(\pi)$) and active infinitely often. Obviously A_i is bad and $r \geq r_i$. If $r \neq r_\ell$ for all bad set A_ℓ, then $r = r_j$ for some other A_j that is active infinitely often. Thus A_j has to be ugly. But then by Lemma 6.10 $A_i \subseteq A_j$ and thus $r_i > r_j = r$. But $r \geq r_i$. Q.E.D.

Let $r = \min\{r_\ell : A_\ell \text{ is bad}\}$. To show that Player 1 wins π' it suffices to prove that there is an odd priority occurring infinitely often in π' which, in case there exists a bound q on all priorities appearing in π, is smaller than $2kq + 2r + 2$.

Notice that for any ugly set A_i, the sequence $(p_{in})_{n\in\omega}$ stabilises at some value p_i. Let $p = \max\{p_i : A_i \text{ is ugly}\}$.

We distinguish two cases. First we assume that there exists some priority

$$b \in \mathrm{Inf}(\pi) \setminus \bigcup\{A_i : A_i \text{ is bad}\}.$$

Fix n_0 such that, for all $n > n_0$, $p_{in} = p_i$ for all ugly sets A_i and $a_{in} \neq 2$ for all good sets A_i. Since $b \in \mathrm{Inf}(\pi)$ there exist infinitely many v_n with $n > n_0$ and $\Omega(v_n) = b$. For such v_n we have $\Omega'(v_n, \overline{m}_n, \overline{a}_n) = 2kp + 2r_i + 3$ if there is a set A_i (which has to be ugly) such that $b \in A_i$ and $a_i = 2$.

Otherwise $\Omega'(v_n, \overline{m}_n, \overline{a}_n)$ is odd and $\leq 2b + 1$. Since A_i is ugly and A_ℓ is bad it follows that $A_\ell \subseteq A_i$. Thus, $r_i < r$. Further $p \leq q$. It follows that there exists some odd priority $s \leq \max(2kp + 2r_i + 3, 2b + 1) < 2kq + r + 2$ that appears in π' infinitely often.

Now we consider the other case: every $b \in \mathrm{Inf}(\pi)$ is contained in some bad set $A_{i(b)}$. Let $A_1, \ldots, A_\ell$ be the bad sets. Without loss of generality, we assume that A_1 is a maximal bad set, i.e., $A_1 \not\subseteq A_i$ for $i = 2, \ldots, \ell$. Since A_1 is a strict subset of $\mathrm{Inf}(\pi)$, we can fix a priority $d \in \mathrm{Inf}(\pi) \setminus A_1$; since any priority $d \in \mathrm{Inf}(\pi)$ is contained in some bad set, we can assume that $d \in A_2$. Further, by the maximality of A_1, we can fix priorities $e_2, \ldots, e_\ell$ where $e_i \in A_1 \setminus A_i$.

We consider a suffix of π that starts at a position where

- all sequences $(p_{in})_{n\in\omega}$ that stabilise at some value p_i have already reached that value;
- all good sets A_i have become inactive for good (i.e. $a_i \neq 2$),
- in the visiting sequence for A_1 the prefixes $p(e_1), \ldots p(e_\ell)$ have already been completed.

Note that A_1 is updated infinitely often, and between any two consecutive points in this suffix at which $\mathrm{up}_1 = 2$ all priorities $e_2, \ldots, e_\ell$ are seen at least once. Since the priority d appears infinitely often in π and A_2 is updated infinitely often, we are going to see infinitely many points v_{n_0} in the considered suffix of π for which $\Omega(v_{n_0}) = d$ and $a_1 = 0$ (since a_1 is reset with an update of A_2). Since a_1 increases to 2 infinitely often, there are infinitely many tuples $n_0 < n_1 < n_2$ such that $a_1 = i$ at all positions v_n with $n_i \leq n < n_{i+1}$ and $a_1 = 2$ at v_{n_2}.

By definition $\mathrm{up}_1 = 2$ at v_{n_1} and v_{n_2} and there cannot be any updates on priority d between v_{n_1} and v_{n_2}, as then a_1 would be reset to 0. By our choice of the considered suffix of π, there are updates on all $e_2, \ldots, e_\ell$ between v_{n_1} and v_{n_2}. Therefore, for any bad set A_j that contains d, we have that $a_j < 2$ between position v_{n_2} and the first position v_n with $\Omega(v_n) = d$ that comes after v_{n_2}. This is the case because between v_{n_1} and v_{n_2} the value a_j was reset to 0 by the update of the visiting sequence for A_1 by priority e_j, and since then it has not increased by more than 1 since there was no update on priority d.

Let us now consider the new priority at v_n. Since all bad sets A_j containing d are inactive, we have the same situation as in the first case: $\Omega'(v_n, \overline{m}_n, \overline{a}_n) = 2kp + 2r_i + 3$ if there is a set A_i (which has to be ugly) such that $d \in A_i$ and $a_i = 2$. Otherwise $\Omega'(v_n, \overline{m}_n, \overline{a}_n)$ is odd and $\leq 2d+1$. Since A_i is ugly and A_ℓ is bad it follows that $A_\ell \subseteq A_i$ and thus $r_i < r_\ell = r$. Further $p \leq q$.

There are infinitely many such positions v_n. Thus there must exist some odd priority $s \leq \max(2kp + 2r_i + 3, 2d + 1) < 2kq + r + 2$ that appears in π' infinitely often. Q.E.D.

Of course, the same arguments apply to the case where $\mathcal{F}_1$ is finite.

Corollary 6.12. Let $(\mathcal{F}_0, \mathcal{F}_1)$ be a Muller winning condition such that either $\mathcal{F}_0$ or $\mathcal{F}_1$ is finite. Then every Muller game with this winning condition is determined via FAR memory.

6.4 Max parity games with bounded moves

We say that an arena (G, Ω) has *bounded moves* if there is a natural number d such that $|\Omega(v) - \Omega(w)| \leq d$ for all edges (v, w) of G.

We have shown in Proposition 4.1 that, in general, winning strategies for max-parity games require infinite memory, but we do not know whether max-parity games are determined via FAR-memory.

For max-parity games with bounded moves, it is still the case that winning strategies may require infinite memory, but now we can prove determinacy via FAR-memory.

Proposition 6.13. There exist max-parity games with bounded moves whose winning strategies require infinite memory.

Proof. Consider a (solitaire) max-parity game with a single node v_0 of priority 0 from which Player 0 has, for every odd number $2n+1$, the option to go through a cycle C_n consisting of nodes with priorities $2, 4, \dots, 2n, 2n+1, 2n, 2n-2, \dots, 4, 2$ and back to the node v_0. All these cycles intersect only at v_0. Clearly Player 0 has a winning strategy, namely to go successively through cycles $C_1, C_2, \dots$ with the result that there is no maximal priority occurring infinitely often. However, if Player 0 moves according to a finite-memory strategy then only finitely many cycles will be visited and there is a maximal n such that the cycle C_n will be visited infinitely often. Thus the maximal priority seen infinitely often will be $2n+1$ and Player 0 loses. Q.E.D.

Lemma 6.14. Let π be a play of a max-parity game $\mathcal{G}$ with bounded moves such that infinitely many different priorities occur in π. Then $\max(\mathrm{Inf}(\pi))$ does not exist, so π is won by Player 0.

Proof. Assume that moves of $\mathcal{G}$ are bounded by d and $\mathrm{Inf}(\pi) \neq \varnothing$ and let q be any priority occurring infinitely often on π. Since infinitely many different priorities occur on π it must happen infinitely often that from a position with priority q the play eventually reaches a priority larger than $q+d$. Since moves are bounded by d, this means that on the way the play has to go through at least one of the priorities $q+1, \dots, q+d$. Hence at least one of these priorities also occurs infinitely often, so q cannot be maximal in $\mathrm{Inf}(\pi)$. Q.E.D.

Theorem 6.15. Every max-parity game with bounded moves can be reduced via a one-dimensional FAR-memory to a parity game. Hence max-parity games are determined via strategies with one-dimensional FAR-memory.

Proof. The FAR-memory simply stores the maximal priority m that has been seen so far. To reduce a max-parity game $\mathcal{G}$ with bounded moves, via this memory, to a parity game $\mathcal{G}'$ we define the priorities of $\mathcal{G}'$ by

$$\Omega'(v, m) = 2m - \Omega(v).$$

Let π be a play of $\mathcal{G}$ and let π' be the extended play in $\mathcal{G}'$. We distinguish two cases. First, we assume that on π the sequence of values for m is unbounded. This means that infinitely many different priorities occur on π, so by Lemma 6.14, Player 0 wins π. But since $m \leq \Omega'(v, m)$ and m never stabilises there is no priority that occurs infinitely often on π, so π' is also won by Player 0.

In the second case there exists a suffix of π on which m remains fixed on the maximal priority of π. In that case $\mathrm{Inf}(\pi)$ is a non-empty subset of $\{0, \ldots, m\}$ and $\mathrm{Inf}(\pi')$ is a non-empty subset of $\{m, \ldots, 2m\}$. Further, $\Omega'(v, m)$ is even if, and only if $\Omega(v)$ is even, and $\Omega'(v_1, m) < \Omega'(v_2, m)$ if, and only if, $\Omega(v_1) > \Omega(v_2)$. Thus, $\min(\mathrm{Inf}(\pi'))$ is even if, and only if, $\max(\mathrm{Inf}(\pi))$ is even. Hence π is won by the same Player as π'. Q.E.D.

7 Conclusion

We have introduced a new memory structure for strategies in infinite games, called FAR-memory, which is appropriate for games with infinitely many priorities and which generalises the LAR-memory for finitary Muller games. We have shown that there are a number of infinitary Muller winning conditions with the following two properties.

(1) There exist Muller games with these winning conditions all whose winning strategies require infinite memory.

(2) All Muller games with such winning conditions can be reduced via FAR-memory to parity games. Therefore all these games are determined via FAR-memory.

The class of these Muller conditions includes: **(1)** downward cones, **(2)** singleton conditions, **(3)** finite unions of upwards cones, and **(4)**winning conditions with finitely many winning sets.

Further we have shown that the same property holds for max-parity games with bounded moves. It is open whether arbitrary max-parity games are determined via FAR-memory. It would also be desirable to obtain a complete classification of the infinitary Muller conditions with this property.

References

[Ba$_0$Vo05] F. Baader & A. Voronkov, eds. *Logic for Programming, Artificial Intelligence, and Reasoning, 11th International Conference, LPAR 2004, Montevideo, Uruguay, March 14–18, 2005, Proceedings*, *Lecture Notes in Computer Science* 3452. Springer, 2005.

[Be$_2$+06] D. Berwanger, A. Dawar, P. Hunter & S. Kreutzer. DAG-width and parity games. In [Du$_1$Th06, pp. 424–436].

[Be$_2$Gr$_0$04] D. Berwanger & E. Grädel. Fixed-point logics and solitaire games. *Theory of Computing Systems* 37(6):675–694, 2004.

[Be$_2$Gr$_0$05] D. Berwanger & E. Grädel. Entanglement – A measure for the complexity of directed graphs with applications to logic and games. In [Ba$_0$Vo05, pp. 209–223].

[$Bo_4Se_1Wa_5$03] A. Bouquet, O. Serre & I. Walukiewicz. Pushdown games with unboundedness and regular conditions. In [$Pa_3Ra_1$03, pp. 88–99].

[$Br_0$02] J.C. Bradfield, ed. *Computer Science Logic, 16th International Workshop, CSL 2002, 11th Annual Conference of the EACSL, Edinburgh, Scotland, UK, September 22–25, 2002, Proceedings*, *Lecture Notes in Computer Science* 2471. Springer, 2002.

[Ca_0Du_0Th02] T. Cachat, J. Duparc & W. Thomas. Solving pushdown games with a Σ_3 winning condition. In [$Br_0$02, pp. 322–336].

[Du_1Th06] B. Durand & W. Thomas, eds. *STACS 2006: 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23–25, 2006, Proceedings*, *Lecture Notes in Computer Science* 3884. Springer, 2006.

[$DzJu_0Wa_5$97] S. Dziembowski, M. Jurdziński & I. Walukiewicz. How much memory is needed to win infinite games? In [$Wi_1$97, pp. 99–110].

[$EmJu_1$91] A. Emerson & C. Jutla. Tree automata, mu-calculus and determinacy. In [$Si_2$91, pp. 368–377].

[$EmJu_1Si_3$01] A. Emerson, C. Jutla & P. Sistla. On model checking for the μ-calculus and its fragments. *Theoretical Computer Science* 258(1–2):491–522, 2001.

[$Gi_0$04] H. Gimbert. Parity and exploration games on infinite graphs. In [$Ma_4Ta_1$04, pp. 56–70].

[$Gr_0$05] E. Grädel. Finite model theory and descriptive complexity. In [Gr_0+07, pp. 125–230].

[Gr_0+07] E. Grädel, P.G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M.Y. Vardi, Y. Venema & S. Weinstein, eds. *Finite Model Theory and Its Applications.* Texts in Theoretical Computer Science, Springer, 2007.

[$Gr_0Wa_5$06] E. Grädel & I. Walukiewicz. Positional determinacy of games with infinitely many priorities. *Logical Methods in Computer Science* 2(4:6):1–22, 2006.

[$Gu_1Ha_2$82] Y. Gurevich & L. Harrington. Trees, automata and games. In [Le_3+82, pp. 60–65].

[$\mathrm{Ho_0Lo_3}$13] E.W. Hobson & A.E.H. Love, eds. *Proceedings of the Fifth International Congress of Mathematicians, Vol. 2.* Cambridge University Press, 1913.

[$\mathrm{Hu_0So_2}$03] W.A. Hunt, Jr. & F. Somenzi, eds. *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8–12, 2003, Proceedings*, Lecture Notes in Computer Science 2725. Springer, 2003.

[$\mathrm{Ju_0}$98] M. Jurdziński. Deciding the winner in parity games is in UP $\cap$ co-UP. *Information Processing Letters* 68(3):119–124, 1998.

[$\mathrm{Ju_0}$00] M. Jurdziński. Small progress measures for solving parity games. In [$\mathrm{Re_0Ti_0}$00, pp. 290–301].

[$\mathrm{Kl_0}$94] N. Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Annals of Pure and Applied Logic* 69(2–3):243–268, 1994.

[$\mathrm{Le_3}$+82] H.R. Lewis, B.B. Simons, W.A. Burkhard & L. Landweber, eds. *Proceedings of the fourteenth annual ACM symposium on Theory of computing 1982, San Francisco, California, United States, May 05–07, 1982.* ACM Press, 1982.

[$\mathrm{Ma_4Ta_1}$04] J. Marcinkowski & A. Tarlecki, eds. *Computer Science Logic, 18th International Workshop, CSL 2004, 13th Annual Conference of the EACSL, Karpacz, Poland, September 20–24, 2004, Proceedings*, Lecture Notes in Computer Science 3210. Springer, 2004.

[$\mathrm{Ma_7}$75] D.A. Martin. Borel determinacy. *Annals of Mathematics* 102(2):363–371, 1975.

[$\mathrm{Mo_4}$91] A. Mostowski. Games with forbidden positions. Technical Report 78, Instytut Matematyk, University of Gdansk, 1991.

[Ob03] J. Obdrzalek. Fast mu-calculus model checking when tree-width is bounded. In [$\mathrm{Hu_0So_2}$03, pp. 80–92].

[Ob06] J. Obdrzalek. DAG-width – connectivity measure for directed graphs. In [$\mathrm{St_2}$06, pp. 814–821].

[$\mathrm{Pa_3Ra_1}$03] P.K. Pandya & J. Radhakrishnan, eds. *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December*

15–17, 2003, Proceedings, *Lecture Notes in Computer Science* 2914. Springer, 2003.

[$\mathrm{Re}_0\mathrm{Ti}_0$00] H. Reichel & S. Tison, eds. *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France, February 2000, Proceedings*, *Lecture Notes in Computer Science* 1770. Springer, 2000.

[$\mathrm{Si}_2$91] M. Sipser, ed. *Proceedings, 32nd Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 1991.

[$\mathrm{St}_2$06] C. Stein, ed. *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22–26, 2006*. ACM-SIAM, ACM Press, 2006.

[$\mathrm{Wi}_1$97] G. Winskel, ed. *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science (LICS '97), Warsaw, Poland, June 29–July 2, 1997*. IEEE, 1997.

[Ze13] E. Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachpiel. In [$\mathrm{Ho}_0\mathrm{Lo}_3$13, pp. 501–504].

[Zi98] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science* 200(1–2):135–183, 1998.

Logics of Imperfect Information: Why Sets of Assignments?[*]

Wilfrid Hodges

Queen Mary
University of London
Mile End Road
London E1 4NS, United Kingdom
w.hodges@qmul.ac.uk

Abstract

We look for an intuitive reason why the Tarski-style semantics of logics of imperfect information (the author's trump semantics) needs sets of assignments where Tarski's semantics for standard logic uses single assignments. To find one, we generalise the semantics to a certain class of games. In this setting, sets of assignments arise naturally; under perfect information one can reduce from sets of assignments to single assignments by choice and gluing. Also truth values can be real numbers, and the number of players can be any natural number ≥ 2; so bivalence and the dialogue between two players are not needed for trump-style semantics.

1 The source of the question

In 1961 Leon Henkin [He$_1$61] extended first-order logic by adding partially ordered arrays of quantifiers. He proposed a semantics for sentences φ that begin with quantifier arrays of this kind: φ is true in a structure A if and only if there are a sentence φ^+ and a structure A^+ such that:

- φ^+ comes from φ by removing each existential quantifier $\exists y$ in the partially ordered prefix, and replacing each occurrence of the variable y by a term $F(\bar{x})$ where $\bar{x}$ are the variables universally quantified 'before' $\exists y$ in the quantifier prefix (so that the new function symbols F are *Skolem function symbols*),
- A^+ comes from A by adding functions to interpret the Skolem function symbols in φ^+, and

[*] My warm thanks to the organisers of 'Interactive Logic: Games and Social Software' for a very stimulating meeting, and for giving me the opportunity to present this paper; and to the organisers of the programme 'Logic and Algorithms' at the Isaac Newton Institute in Cambridge, where the paper was written up. Also I thank the referee for helpful suggestions towards improving the presentation.

- φ^+ is true in A^+.

For example the sentence

$$\begin{array}{c}(\forall x)(\exists y)\\(\forall z)(\exists w)\end{array}\ \psi(x,y,z,w) \tag{1.1}$$

is true in A if and only if there are functions F^A, G^A such that

$$(A, F^A, G^A)\ \models\ (\forall x)(\forall z)\psi(x, F(x), z, G(z)) \tag{1.2}$$

where F, G stand for F^A, G^A respectively. Jon Barwise commented seventeen years later:

> ...the meaning of a branching quantifier expression of logic like:
>
> $$\left.\begin{array}{ccc}\forall x & — & \exists y\\ \\ \forall z & — & \exists w\end{array}\right> \psi(x,y,z,w) \tag{1.3}$$
>
> cannot be defined inductively in terms of simpler formulas, by explaining away one quantifier at a time. Rather, the whole block
>
> $$\left.\begin{array}{ccc}\forall x & — & \exists y\\ \\ \forall z & — & \exists w\end{array}\right> \tag{1.4}$$
>
> must be treated at once. [Ba$_6$78][1]

He offered a proof of what he called 'a precise version of this claim'. Unfortunately his proof proves much less than he said. It shows only that truth for Henkin's sentences is not an inductive verifiability relation in the sense of Barwise and Moschovakis [Ba$_6$Mo$_1$78]. The key point is that the inductive clauses can't be first-order; but this is hardly surprising.

2 Separating out the problems

With hindsight we can see that there are at least three problems that stand in the way of giving an inductive definition of truth for sentences with partially ordered quantifier arrays.

The first problem is that the definition needs to describe the effect of adding a single quantifier at the lefthand end of a formula. But for partially ordered quantifier arrays there may be several lefthand ends, and it makes a difference where we add the quantifier. The sentence (1.3) behaves quite differently from

$$\left.\begin{array}{ccccc}\forall z & — & \forall x & — & \exists y\\ & & & & \exists w\end{array}\right> \psi(x,y,z,w) \tag{1.5}$$

[1] We use ψ in place of Barwise's A to avoid a clash of notation, and also number the formulas.

I don't know that anybody has thought seriously about this problem; it needs a subtler notion of substitution than we are used to in logic. But in any case Jaakko Hintikka showed how to sidestep it by using a notation that shakes Henkin's formulas down into a linear form. The formula (1.1) above becomes

$$(\forall x)(\exists y)(\forall z/\exists y)(\exists w/\forall x)\psi(x,y,z,w) \tag{1.6}$$

in Hintikka's notation. (Hintikka also introduced a dual notion of falsehood in terms of Skolem functions for the universal quantifiers. Thus the slash $/\exists y$ in (1.6) expresses that the function for z is independent of y.) This linear notation forms the syntax of the 'Independence-Friendly' IF logic of Hintikka and Sandu [$\mathrm{Hi_1Sa_4}$96]. Hintikka also pointed out that the Skolem functions can be regarded as strategies in a game between $\forall$ and $\exists$; the resulting games form the game semantics for IF logic.

The second problem is that a Skolem function for an existential quantifier is a function of the preceding universally quantified variables, but not of the preceding existentially quantified ones. But for example the formula

$$(\exists w/\forall x)\psi(x,y,z,w) \tag{1.7}$$

gives no indication whether the variables y and z are going to be universally or existentially quantified; so from the formula alone we don't know what information can be fed into the Skolem functions for it.

In [$\mathrm{Ho_1}$97a] I sidestepped this second problem by replacing the Skolem functions by general game strategies. Unlike a Skolem function, a strategy for a player can call on previous choices of either player. For games of perfect information this is a distinction without a difference; if player $\exists$ has a winning strategy using the previous choices of both players, then player $\exists$ has a winning strategy that depends only on the previous choices of player $\forall$. But for games of imperfect information, such as we have here, it makes a difference. I proposed marking the difference between Hintikka's games and mine by dropping the quantifiers after the slashes, and writing for example $(\forall z/y)$ where Hintikka writes $(\forall z/\exists y)$. In what follows we refer to the logic with my notation and the general game semantics as *slash logic*. During recent years many writers in this area (but never Hintikka himself) have transferred the name 'IF logic' to slash logic, often without realising the difference. Until the terminology settles down, we have to beware of examples and proofs that don't make clear which semantics they intend.

Given a solution to the third problem (below), solving the second problem for IF logic with the Skolem function semantics is tiresome but not difficult. The solution is to give several different interpretations for each formula, one for each guess about which free variables are going to be quantified existentially. As we add the quantifiers, we discard the wrong guesses—a bit like Cooper storage. Details are in [Ho$_1$97b]. I believe a similar trick should deal with the first problem when it has been correctly posed.

There remains the third problem, which is to find an inductive truth definition for slash logic. The paper [Ho$_1$97a] gave the trump semantics, which solves this problem.

It turned out that the main idea needed was to think of formulas as being satisfied not by assignments to their variables, but by *sets of assignments*. Why sets of assignments? Mathematically the idea is natural enough, but we can hardly see by intuition that it will work. For a while I almost convinced myself that an inductive truth definition for slash logic would need sets of sets of assignments, sets of sets of sets of assignments, and so on up the hierarchy of finite types.

An intuitive answer to the question 'Why sets of assignments?' could do marvels for making the inductive semantics of these sentences more appealing. But it's hardly clear where to look for an intuition. One obvious approach is to try generalising the semantics to other logics, in order to see what is needed where. But in the last ten years this hasn't happened. The semantics has been extended, but only to logics with broadly the same features as Henkin's original.

So we have to look elsewhere. One suggestion runs as follows. The trump semantics for formulas of slash logic was found by starting from a game semantics on sentences. The passage from sentences to games to trumps is too complex to support any strong intuition. So we should try to separate the games from the trump semantics. There are two natural ways to do this. The first way is to discard the games altogether and find a direct motivation for the trump semantics. Jouko Väänänen has made good progress in this direction [VäHo$_1\infty$].

The second way is to abandon the formulas and work directly with the games. This is the purpose of this paper.

3 The programme

We aim to extract the game-theoretic content of the games in [Ho$_1$97a], and extend it to a class of games that has no intrinsic connection with formulas or truth values. Finding the trump semantics was a matter of extending the truth values on sentences to semantic values on formulas, in such a way that the values on formulas could be built up by induction on complexity. We aim to do the same but in a purely game-theoretic setting: we define values on games, and we extend these values to values on subgames, again with the aim of defining these values by induction.

This description is too open-ended for comfort. Since we've thrown away the connection with truth, what counts as a correct extension to subgames?

Fortunately we know the formal core of the Tarski truth definition; Section 4 below describes it in terms of fregean values. This formal description carries over straightforwardly from formulas to games, as soon as we have said what subgames are and what the value of a game is. Section 5 will describe the games and their subgames. Section 6 will propose suitable values for games. (There might be better choices here that I didn't think of.) Then Sections 7 and 8 carry out the extension to fregean values, first for games of perfect information and then under imperfect information.

As hoped, the fregean value of a subgame is in terms of sets of assignments rather than single assignments. But also we can see a game-theoretic reason for this. The following summary will perhaps make more sense at the end of the paper, but I hope it conveys something already at this stage.

The value of a game is defined in terms of the existence of certain strategies for the players. The definition of values by induction on subgames builds up these strategies. Now one familiar move in building up a strategy σ for a player p, at a place where another player p' is about to move, is to find a strategy σ_a corresponding to each possible choice a by p'. We build σ as follows: player p waits to see what choice a player p' makes, and then proceeds with σ_a. But under conditions of imperfect information p may not know what choice p' is making. (Or to say the same thing in terms of information partitions, the composite strategy σ may give different answers on data items in the same partition set.) So this kind of gluing is blocked. The consequence is that we can give player p this strategy σ only if we know that it works uniformly for all choices that player p' can make. In other words, the information that we have to carry up from the subgames is not that certain data states allow strategies, but that certain *sets* of data states allow the same strategy. In short, we need to go up one type level in order to sidestep the fact that we can't in general glue strategies together.

4 Fregean values

This section summarises without proofs the main results in [Ho$_1\infty$].

Suppose $\mathbb{E}$ is a set of objects called *expressions* (for example the formulas of a logic). We assume expressions can have parts that are also expressions. We write $F(\xi_1, \ldots, \xi_n)$ for an 'expression with holes', that gives an an expression $F(e_1, \ldots, e_n)$ when suitable expressions $e_1, \ldots, e_n$ are put in the holes, putting e_i in hole ξ_i for each i.

We assume given a family $\mathbb{F}$ of such 'frames' $F(\xi_1, \ldots, \xi_n)$, with four properties:

1. $\mathbb{F}$ is a set of nonempty partial functions on $\mathbb{E}$. ('Nonempty' means their domains are not empty.)

2. (Nonempty Composition) If $F(\xi_1, \ldots, \xi_n)$ and $G(\eta_1, \ldots, \eta_m)$ are

frames, $1 \leqslant i \leqslant n$ and there is an expression

$$F(e_1, \ldots, e_{i-1}, G(f_1, \ldots, f_m), e_{i+1}, \ldots, e_n),$$

then $F(\xi_1, \ldots, \xi_{i-1}, G(\eta_1, \ldots, \eta_m), \xi_{i+1}, \ldots, \xi_n)$ is a frame.

3. (Nonempty Substitution) If $F(e_1, \ldots, e_n)$ is an expression, $n > 1$ and $1 \leqslant i \leqslant n$, then

$$F(\xi_1, \ldots, \xi_{i-1}, e_i, \xi_{i+1}, \ldots, \xi_n)$$

is a frame.

4. (Identity) There is a frame $1(\xi)$ such that for each expression e, $1(e) = e$.

Assume also that $S \subseteq \mathbb{E}$. We refer to the expressions in S as *sentences*, since this is what they are in most applications to logics. Assume that a function μ with domain S is given. The function μ gives a 'value' to each sentence; we make no assumptions at all about what kinds of object these values are.

Under these assumptions, we define a relation $\equiv$ on $\mathbb{E}$ as follows. (It expresses that two expressions make the same contribution to the μ-values of sentences containing them.) For all expressions e and f, $e \equiv f$ if and only if

(a) for each frame $F(\xi)$ with one variable, $F(e)$ is in S if and only if $F(f)$ is in S;

(b) whenever $F(e)$ and $F(f)$ are in S, $\mu(F(e)) = \mu(F(f))$.

Then $\equiv$ is an equivalence relation. If distinct values are assigned to the distinct equivalence classes, then we call the value $|e|$ assigned to the equivalence class of e the *fregean value* of e. (Again the nature of these values is unimportant; all that matters is whether or not two expressions have the same fregean value.)

Lemma 4.1. Suppose that for every expression e there is a frame $F(\xi)$ such that $F(e)$ is a sentence. Then for each frame $G(\xi_1, \ldots, \xi_n)$ there is a function h_G such that for all expressions $e_1, \ldots, e_n$ such that $G(e_1, \ldots, e_n)$ is an expression,

$$|G(e_1, \ldots, e_n)| = h_G(|e_1|, \ldots, |e_n|).$$

In a common turn of phrase, Lemma 4.1 says that fregean values are 'compositional'. The function h_G is called the *Hayyan function* of G in [Ho$_1\infty$]. (The name 'fregean' was suggested by some passages in Frege's *Grundlagen der Arithmetik*; Hayyan functions are named after the 14th century linguist Abu Hayyan al-Andalusi, who argued that such functions must exist.)

Lemma 4.2. Let e and f be sentences. If $e \equiv f$ then $\mu(e) = \mu(f)$. Hence there is a function r defined on the fregean values of sentences, such that for every sentence e, $\mu(e) = r(|e|)$.

The message of these two lemmas together is that the function μ, which is quite arbitrary, can always be defined through an inductive definition of fregean values, where the induction is on the complexity of expressions. The fregean values of atomic expressions have to be given directly as the base case. The Hayyan functions take care of the inductive steps, and finally the function r reads off μ from the fregean values of the sentences. It turns out that in standard examples of truth definitions we can take r to be the identity. One such case is where E is the set of formulas of slash logic, S is the set of sentences, μ is the assignment of truth values to sentences as given by the game semantics, and the fregean values are the values given to formulas by the trump semantics.

The intended applications of this machinery were languages. But nothing prevents us from carrying them over to games. The set S will consist of the games and the set $\mathbb{E}$ will consist of the subgames of these games (in some suitable sense). We give each game G a 'value' $\mu(G)$ in terms of the effects of strategies of the players, and then we compute fregean values for the subgames. This will yield an inductive definition for the values $\mu(G)$, working directly on the games without any intervention of formulas. In this setting we can test the effect of moving from games of perfect information to games of imperfect information. Does it move the values up a type?

5 The games

The first step in our programme is to define the games and the subgames so that the assumptions of Section 4 hold. The requirements are fairly strong. The games have to be constructed inductively from their subgames. The notion of substituting one subgame for another has to make sense. So does the notion of imperfect information. Already the games start to look a little like formulas. But we can discard the notion that there are just two players, and we don't need the notions of winning and losing. The players need not be in competition with each other.

I think we need the following ingredients.

- First and trivially, there must be more than one player. (Otherwise the notion of imperfect information becomes degenerate.) We write $\mathcal{P}$ for the set of players.

- Second, as the game proceeds, the players build up a bank of data (corresponding to the assignment of elements to variables). It's enough to assume there is a set $\mathcal{Q}$ of *questions*, each of which has a nonempty set of answers. A *data state* is a function defined on a set of questions, taking each of these questions to one of its answers. We write $\mathcal{D}$ for the set of data states.

- Third, some of the choices of the players are structural; they have no effect on the data state, but they control who moves next, what the criteria are for deciding the payoffs, what information can be fed into strategies, and so forth. Thus at any stage of the play a data state s and a sequence $\bar{c}$ of structural moves have been built up. The information fed into the play by the sequence $\bar{c}$ determines a 'subgame' which is to be played 'at' the data state s.

- Fourth, the fact that the same subgame can be played at different data states allows the possibility that the player who moves at this subgame may have incomplete information about the data state.

- Fifth, the players can move to subgames only finitely often in a play. Eventually they reach an atomic subgame; when they do, the payoff to each player is determined by the subgame and the data state. There is no need to assume that the payoffs are wins or losses; real number values will suffice, and they need not add up to zero.

- Sixth, for each subgame H there is an associated set of questions $\mathrm{m}(H)$, which we can call the *matter* of H, with the property that H can be played as soon as we are given a data state s which answers all questions in $\mathrm{m}(H)$. (This corresponds to the free variables of a formula.) We need an assumption like this in order to make sense of substitution of subgames.

The following definitions are meant to give shape to these ingredients. They are strongly influenced by Parikh's paper [Pa₅83].

We assume given the set $\mathcal{P}$ of players, the set $\mathcal{Q}$ of questions and the set $\mathcal{D}$ of data states, as above. Given a finite subset W of $\mathcal{Q}$, we write $\mathcal{D} \restriction W$ for the set of all data states whose domain is W, and $\mathbb{R}^{\mathcal{D}\restriction W}$ for the set of all functions from $\mathcal{D} \restriction W$ to the set $\mathbb{R}$ of real numbers. If s is a data state, q is a question and a is an answer to q, we write $s(q/a)$ for the data state whose domain is domain(s) $\cup\{q\}$, and which agrees with s everywhere except that $s(q/a)(q) = a$.

We define inductively the set of *game parts*. Formally, the game parts will be finite sequences. Later we will explain how they are played.

(α) For every finite $W \subseteq \mathcal{Q}$ and every function $f : \mathcal{P} \to \mathbb{R}^{\mathcal{D}\restriction W}$, $\langle W, f\rangle$ is a game part; its *matter* is W. Game parts of this form are *atomic.*

(β) For every finite set $W \subseteq \mathcal{Q}$, every player $p \in \mathcal{Q}$, every partition π of $\mathcal{D} \restriction W$, every question $q \in \mathcal{Q} \setminus W$ and every game part J with matter $\subseteq W \cup \{q\}$, there is a game part $\langle W, p, \pi, q, J\rangle$ whose matter is W.

(γ) For every finite set $W \subseteq \mathcal{Q}$, every player $p \in \mathcal{P}$, every partition π of $\mathcal{D} \restriction W$ and every nonempty set X of game parts with matter $\subseteq W$, there is a game part $\langle W, p, \pi, X \rangle$ whose matter is W.

For each game part H with matter W and each data state s with domain $\supseteq W$, H is played at s as follows:

(α) If H is $\langle W, f \rangle$ then there is an immediate payoff of $f(p)(s \restriction W)$ to each player p.

(β) If H is $\langle W, p, \pi, q, J \rangle$, then player p moves by choosing an answer a to the question q, and the play continues as a play of J at the data state $s(q/a)$.

(γ) If H is $\langle W, p, \pi, X \rangle$, then player p moves by choosing a game part $J \in X$, and the play continues as a play of J at the data state s.

Each game part H determines a labelled tree $T(H)$ branching downwards. If H is $\langle W, f \rangle$ then $T(H)$ has a single node, labelled with H. If H is $\langle W, p, \pi, q, J \rangle$ then $T(H)$ is $T(J)$ with a single node added at the top, carrying the label H. If H is $\langle W, p, \pi, X \rangle$ then the top node of $T(H)$ carries the label H, and the successors of the top node are the top nodes of copies of the trees $T(J)$, one for each $J \in X$.

A *game* is a game part with empty matter. We say that a game part H is a *subgame of* a game part G if H occurs as a label on a node of $T(G)$. Note that a game part G can have several occurrences of a game part H in it.

Suppose a play of a game part G at a state s is in progress, and the players have just reached an occurrence of the subgame H of G. Then the choices of the players consist of: (1) the sequence $\bar{c}$ of subgames that label the nodes as one travels down $T(G)$ from the top to the relevant occurrence of H—the sequence lists the game parts chosen at subgames of the form (γ) in earlier moves of the play; (2) a data state t representing s together with the choices made at subgames of the form (β) in earlier moves. We call the pair $(\bar{c}, t)$ a *position* in the play. The pair $(\bar{c}, s)$ determines the domain of t (namely, the union of the domain of s and the set of questions answered at moves reported in $\bar{c}$); we call this domain the *current domain* at $(\bar{c}, s)$. The sequence $\bar{c}$ also determines the player who will move next; we call this player the *current player* at $\bar{c}$. The *current game part* at the position $(\bar{c}, t)$ is the final game part H of $\bar{c}$. If this game part is of the form $\langle W, p, \pi, \ldots \rangle$, then p and π are respectively the *current player* and the *current partition* at $\bar{c}$. Here the current game part, player and partition depend only on $\bar{c}$, and the current domain depends only on $\bar{c}$ and s; but since the position $(\bar{c}, t)$ determines s, we can speak of any of these things as being *current* at the position $(\bar{c}, t)$.

A *strategy* for a player p in a game part G at a data state s is a family σ of functions $\sigma_{\bar{c}}$ indexed by the sequences $\bar{c}$ such that p is the current player at $\bar{c}$. For each position $(\bar{c}, t)$ where p is the current player, $\sigma_{\bar{c}}(t)$ is a possible move for p in the current subgame. There is some redundancy here, because a strategy is defined at positions in the game which could never be reached if the strategy was followed; but this redundancy will never matter and it would be a nuisance to exclude it. Note that the strategy σ depends on s and not just on G; the dependence on s will be important below.

So far the partitions have played no role. Their purpose is to restrict the allowed strategies, as follows. We say that a strategy σ for player p in game part G at s is *admissible at* $\bar{c}$ if for all data states t, u with domain the current domain W at $\bar{c}$, if

> $s \restriction W$ and $t \restriction W$ lie in the same partition set of the current partition at $\bar{c}$,

then

$$\sigma_{\bar{c}}(s) = \sigma_{\bar{c}}(t).$$

We say that a strategy σ for p is *admissible* if it is admissible at all $\bar{c}$ at which p is the current player.

A game G is *of perfect information* if all the partitions appearing in G are trivial, i.e. all their partition sets are singletons. For a game of perfect information, every strategy is admissible. When we discuss games of perfect information, we can ignore their partitions; for example we can write $\langle W, p, \pi, X\rangle$ as $\langle W, p, X\rangle$.

In Sections 7 and 8 below we will sometimes talk of strategies that are restricted to subsets of some set $\mathcal{D} \restriction W$. It makes sense to glue together several such strategies, provided that no two of them are defined on members of the same partition.

If J is a game part occurring in the subgame H, and J' is a subgame with the same matter as J, then we can form a new subgame $H(J'/J)$ by replacing the occurrence of J by an occurrence of J'; the matter of $H(J'/J)$ is the same as that of H. (The notation is a shorthand; there might be other occurrences of J in H, and these stay unchanged.)

It would be possible to substitute J' for J in H even if the matter of J' is not the same as that of J. But suppose the question q is in the matter of J' and not in that of J. Let G be a game where some player chooses in turn answers to the questions in $m(J)$, and then the game continues as J. Then substituting J' for J in G yields a subgame G' whose matter contains q; so this substitution turns a game into a game part that is not a game. We will bar this kind of substitution. In other words, we will consider only substitutions where (a) of Section 4 holds. So the significant question will be when (b) holds too.

Our notion of subgame is not the more familiar one due to Selten [Se$_0$65]. A Selten subgame in our context would be a pair (G, s) where G is a subgame that can be played at data state s. For us it is essential that the same subgame can be played at different data states. This is the game analogue of the fact that a subformula allows different assignments to its variables.

6 Game values

We define the *value* of a game G to a player p, $\mu_p(G)$, to be the supremum of the reals λ such that p has an admissible strategy which ensures that the payoff to p is at least λ. So $\mu_p(G)$ is an element of $\mathbb{R} \cup \{\pm\infty\}$. We can take the *value* $\mu(G)$ of G to be the function taking each player p to $\mu_p(G)$. But in fact all our calculations of values will consider one player at a time.

This is a generalisation of the assignment of truth values to sentences in logic. The logical case is where there are two players, the payoffs are all either 0 or 1, and for any atomic game part at any data state the payoffs to the two players add up to 1. One could generalise in other ways (for example taking values in a complete boolean algebra), but I chose something simple that seems to fit with the habits of game theory.

Now that we have the values of games, we can apply the framework of Section 4. We take $\mathbb{E}$ to be the set of game parts, S to be the set of games and μ to be the value function just defined. Every game part is a subgame of a game, so that the hypothesis of Lemma 4.1 holds. (It would have failed if we allowed the matter of a game part to be infinite, since only finitely many questions get answered during a play.)

Following Section 4 we define a relation $\equiv$ on subgames: $H \equiv J$ if and only if (a) H and J have the same matter, and (b) for every game G containing an occurrence of H, $\mu(G(J/H)) = \mu(G)$. Then $\equiv$ is an equivalence relation on $\mathbb{E}$. If we can identify the equivalence classes, we can label them with fregean values, and then we have a definition of μ by induction on subgames.

7 The extension under perfect information

In this section all games are of perfect information, so that all strategies are admissible. Here the situation is familiar enough to suggest where to look for fregean values.

Definition 7.1. Let G be a game part.

(a) Let p be a player and s a data state with domain $\supseteq m(G)$. Define the *value* of G at s to p, $v_p(G,s)$, by:

$$v_p(G,s) = \sup\{\lambda : p \text{ has a strategy which, when } G \text{ is played at state } s, \text{ guarantees that the payoff to } p \text{ is at least } \lambda\}.$$

(b) We define $v_p(G)$ to be the function with domain $\mathcal{D} \restriction m(G)$, whose value for each s in this set is $v_p(G,s)$.

(c) We define $v(G)$ to be the function with domain $\mathcal{P}$, whose value for each player p is $v_p(G)$.

In the case where G is a game, $v_p(G) = \mu_p(G)$ for each player p, and so $v(G) = \mu(G)$. In the case where G is atomic, there is an immediate payoff to each player for each $s \in \mathcal{D} \restriction m(G)$, and $v(G)$ records these payoffs.

The definition of $v(G)$ depends only on the values $v(G,s)$ where s has domain $m(G)$. But G can be played at data states s with much larger domains. We need to show that for these s the values $v(G,s)$ are determined by $v(G)$. (This is fundamental. If it failed, the values $v(G)$ wouldn't be fregean values obeying the conclusion of Lemma 4.1.)

Lemma 7.2. (Under perfect information.) Suppose a game part G has matter W, and s is a data state with domain $W' \supseteq W$. Let p be a player. Then the value of G at s for p is equal to the value of G at $s \restriction W$ for p.

Proof. Suppose σ is a strategy for p in G at $s \restriction W$ that guarantees p a payoff of at least λ. Let τ be the following strategy for p in G at s: ignore any answers to questions not in W, and use σ. Induction on the complexity of G shows that this is a strategy for p in G at s; the ignored values are never needed, and in particular they make no difference to the payoff. Thus τ guarantees payoff at least λ for p.

Conversely suppose τ is a strategy for p in G at s that guarantees p a payoff of at least λ. Then let σ be the strategy for p in G at $s \restriction W$ that uses τ, filling in the extra values from s. Again σ guarantees payoff at least λ to p. Q.E.D.

Theorem 7.3. (Under perfect information.) Suppose H and H' are game parts with the same matter. Then $H \equiv H'$ if and only if $v(H) = v(H')$.

Proof. We fix a player p. Assuming that $v_p(H) = v_p(H')$, we prove that for every game part G in which H occurs as a subgame, $v_p(G) = v_p(G(H'/H))$. The proof is by induction on the complexity of G. By the lemma, we need only show that $v_p(G,s) = v_p(G(H'/H),s)$ when s has domain $m(G)$.

(α) Suppose first that $G = H$. Then $G(H'/H) = H'$, so the result is immediate.

(β) Suppose that p' is a player, G is $\langle W, p', q, J\rangle$, s is a data state with domain $m(G)$, and $v_p(G, s) = \lambda$. Then for every $\lambda' < \lambda$, p has a strategy for G at s which guarantees p a payoff of at least λ'. We aim to show the same for $G(H'/H)$. There are two cases, according as p' is p or another player.

Suppose first that $p' \neq p$. Then for each $\lambda' < \lambda$ and each possible choice a of p' at G, there is a strategy σ_a for p for J at $s(a/q)$ which guarantees p a payoff of at least λ'.

Now by induction hypothesis $v_p(J) = v_p(J(H'/H))$, so the lemma tells us that $v_p(J, s(a/q)) = v_p(J(H'/H), s(a/q))$ for each a. It follows that for each a, player p has a strategy τ_a for $J(H'/H)$ at $s(a/q)$ which guarantees p a payoff of at least λ'. For each $\lambda' < \lambda$ we can glue these strategies together to produce a strategy τ for p in G at s, namely: Wait for the choice a and then play τ_a. This strategy guarantees p a payoff of at least λ'. Hence again $v_p(G(H'/H), s) \geqslant v_p(G, s)$, and symmetry gives the converse.

The other case, where p' is p, is similar but easier. The strategy σ for p at G chooses an element a to answer q, and then we need only consider $s(a/q)$ for this a, so that no gluing is needed.

(γ) Suppose p' is a player and G is $\langle W, p', q, X\rangle$. Then the argument of case (β) applies with appropriate changes. Note that since the different subgame occurrences have their own strategy functions, there is no need for any gluing in this case.

This proves one direction. For the other, suppose $v_p(H) < v_p(H')$, and choose λ with $v_p(H) < \lambda < v_p(H')$. Then p has no strategy in H that guarantees that for all s, p will get payoff λ. Hence there is some s such that p can't guarantee to get λ, playing at s. Consider the game where some other player chooses assignments to the domain of s, then p picks up and plays H. In this game G player p can't guarantee to get payoff λ, since the other player could play s. But by assumption player p can guarantee to get payoff λ in $G(H'/H)$. Hence $\mu(G(H'/H)) \neq \mu(G)$, so that $H \not\equiv H'$. Q.E.D.

Suppose we restrict to any smaller class of games which is closed under substitution of subgames with the same matter, and under (β) of Section 5. (An example of such a class is where in (γ) we require the set X to be finite. This comes nearest to first-order logic.) Then the entire argument above goes through.

8 The extension under imperfect information

We turn to our major question. What is needed to repair the proof of Theorem 7.3 if we drop the assumption of perfect information?

Lemma 7.2 doesn't survive unaltered, but with a suitable definition of values we can still get the main point of the lemma, which is that the values at data states whose domain is the matter determine the values at all other data states. The proof of Theorem 7.3 also goes through except for one point: in the gluing at case (β), nothing guarantees that the resulting strategy τ is admissible. A little meditation shows that the problem is serious. No information about the existence of separate admissible strategies τ_a for $J(H'/H)$ at $s(a/q)$ is going to guarantee a single admissible strategy for $G(H'/H)$ at s.

So we have to carry up inductively the information that certain *sets* of data states lie within the domains of admissible strategies.

Definition 8.1. Let G be a game part.

(a) Let p be a player, and let W be a finite set of questions $\supseteq m(G)$. Define the *value* of G at a subset S of $\mathcal{D} \restriction W$ to p, $v_p(G, S)$, by:

$$v_p(G,S) = \sup\{\lambda : \begin{array}{l} p \text{ has an admissible strategy which, when} \\ G \text{ is played at any state } s \in S, \text{ guarantees} \\ \text{that the payoff to } p \text{ will be at least } \lambda\}. \end{array}$$

(b) We define $v_p(G)$ to be the function with domain the power set of $\mathcal{D} \restriction m(G)$, whose value for each subset S of $\mathcal{D} \restriction m(G)$ is $v_p(G, S)$.

(c) We define $v(G)$ to be the function with domain $\mathcal{P}$, whose value for each player p is $v_p(G)$.

Then as before, $v(G) = \mu(G)$ whenever G is a game.

Suppose $W' \supseteq W$ and S is a subset of $\mathcal{D} \restriction W'$. We say that s, t in S are *in the same fibre* of S along W if $s \restriction (W' \setminus W) = t \restriction (W' \setminus W)$. This defines an equivalence relation, and its equivalence classes are called the *fibres* of S along W.

Lemma 8.2. Suppose a game part G has matter W, W' is a finite set $\supseteq W$, and S is a set of data states with domain W'. Let p be a player. Then the value of G at S for p is equal to the infimum of the values of G at the fibres of S along W for p.

Proof. (Cf. [Ho$_1$97a, Lemma 7.4].) Let λ be the infimum of the values of G at the fibres of S along W for p. Then for each $\lambda' < \lambda$ and each fibre φ of S along W, there is an admissible strategy σ_φ for p on φ, which guarantees p a payoff of at least λ'. Fixing λ', glue together these strategies on the separate fibres, to get a strategy σ on W. In the definition of admissibility, elements of different fibres never agree off W; so the admissibility of the σ_φ guarantees that σ is admissible. Thus the value of G at S for p is at least λ.

An easier argument in the other direction shows that if the value of G at S for p is at least λ, then the value at each fibre is at least λ too. Q.E.D.

We repeat the theorem, but under imperfect information and with the new definition of v.

Theorem 8.3. Suppose H and H' are game parts with the same matter. Then $H \equiv H'$ if and only if $v(H) = v(H')$.

Proof. We fix a player p. Assuming that $v_p(H) = v_p(H')$, we prove that for every game part G in which H occurs as a subgame, $v_p(G) = v_p(G(H'/H))$. The proof is by induction on the complexity of G. By the lemma, we need only show that $v_p(G, S) = v_p(G(H'/H), S)$ when $S \subseteq \mathcal{D} \restriction m(G)$.

(α) The case where $G = H$ is as before.

(β) Suppose that p' is a player, G is $\langle W, p', \pi, q, J\rangle$, $S \subseteq \mathcal{D} \restriction W$ and $v_p(G, S) = \lambda$. Then for every $\lambda' < \lambda$, p has an admissible strategy for G at S which guarantees p a payoff of at least λ'. We aim to show the same for $G(H'/H)$. There are two cases, according as p' is p or another player.

Suppose first that $p' = p$. Then:

> there is an admissible function σ for p such that p has an admissible strategy for J at $S^\sigma = \{s(\sigma(s)/q) : s \in S\}$ which guarantees p a payoff of at least λ'.

Now by induction hypothesis $v_p(J) = v_p(J(H'/H))$. Hence by the lemma, $v_p(J, S^\sigma) = v_p(J(H'/H), S^\sigma)$. It follows that p has an admissible strategy for $J(H'/H)$ at S^σ which guarantees p a payoff of at least λ'. Combining this strategy with σ, p has an admissible strategy for $G(H'/H)$ at S which guarantees a payoff of at least λ'. Thus $v_p(G) \leqslant v_p(G(H'/H))$, and symmetry gives the converse.

Next, suppose $p' \neq p$. The argument is the same, except that in place of S^σ we use $S^q = \{s(a/q) : s \in S, a \text{ an answer to } q\}$.

(γ) Suppose that p' is a player, G is $\langle W, p', \pi, X\rangle$ and $S \subseteq \mathcal{D} \restriction W$. One can adjust the arguments of case (β) to this case without needing any new ideas.

Now conversely suppose that $\mathrm{m}(H) = m(H') \neq \varnothing$, and for some S with domain $\mathrm{m}(H) = \{q_1, \dots, q_k\}$, $v_p(H, S) < v_p(H', S)$. Then the same inequality must hold for some nonempty intersection of S with a class of the current partition at H; so we can assume that S lies in a single class of this partition. Choose λ, λ' with $v_p(H, S) < \lambda' < \lambda < v_p(H', S)$. Let $f : \mathcal{P} \to \mathbb{R}^{\mathcal{D}\restriction \mathrm{m}(H)}$ be the function that takes each player $p' \neq p$ to the constant function with value 0, and that satisfies

$$f(p)(s) = \begin{cases} \lambda' & \text{if } s \in S, \\ \lambda & \text{otherwise.} \end{cases}$$

Let p_0 be some player other than p, and let G be the game

$$\langle \varnothing, p_0, q_1, \langle \{q_1\}, p_0, q_2, \langle \dots, q_k, \langle \mathrm{m}(H), p, \{H, \langle \mathrm{m}(H), f\rangle\}\rangle \dots \rangle$$

where the missing partitions are all trivial, so that the information is perfect except perhaps within H.

In the game G, player p_0 can use the first k moves to pick an element of S. Then by assumption p has no admissible strategy in H guaranteeing a payoff $\geqslant \lambda'$, and choosing $\langle \mathrm{m}(H), f\rangle$ guarantees p a payoff of only λ'. So $v_p(G) \leqslant \lambda'$. On the other hand p has a strategy for $G(H'/H)$ which guarantees a payoff of at least λ. Namely, if p_0 chooses s in S, then pick H' and play a suitable admissible strategy at S in H'; if p_0 chooses outside S, then choose $\langle \mathrm{m}(H), f\rangle$ and collect λ. Since G is a game, it follows that $H \not\equiv H'$. Q.E.D.

Just as in the previous section, the theorem still holds good if we restrict to a class of games with reasonable closure conditions. I omit details.

9 Conclusion

In the games above, we get fregean values for game parts by assigning values to sets of data states, not to single data states. This is the exact analogue of what happens in the semantics for slash logic as in [Ho$_1$97a]. The proofs make clear why this is the right level: in some sense the argument was *always* about sets of data states rather than data states one at a time—but in the case of perfect information we could disguise this fact by taking the data states one at a time and then gluing.

The games above do look rather like formulas. (I don't know whether they have any other application.) But our arguments show that some features of logical formulas are irrelevant to the fact that fregean values go with sets of data states. In particular the number of players is irrelevant as long as it is at least two. Competition between the players is irrelevant. Truth (as opposed to real number values) is also irrelevant. Last but not least, the information partitions that we allowed are much more general than those that arise from IF or slash logic.

References

[Ba$_6$78] J. Barwise. On branching quantifiers in English. *Journal of Philosophical Logic* 8:47–80, 1978.

[Ba$_6$Mo$_1$78] J. Barwise & Y.N. Moschovakis. Global inductive definability. *Journal of Symbolic Logic* 43(3):521–534, 1978.

[Gu$_0$vBPa$_5\infty$] A. Gupta, J. van Benthem & R. Parikh, eds. *First Indian Conference on Logic and its Relationship with Other Disciplines, Indian Institute of Technology, Bombay, January 8–13, 2005, Proceedings.* Forthcoming.

[He$_1$61] L. Henkin. Some remarks on infinitely long formulas. In *Infinitistic Methods: Proceedings of the Symposium on Foundations of Mathematics, Warsaw, 2–9 September 1959*, pp. 167–183. Pergamon Press and Państwowe Wydawnictwo Naukowe, 1961.

[Hi$_1$Sa$_4$96] J. Hintikka & G. Sandu. Game-theoretical semantics. In [vBtM96, pp. 361–410].

[Ho$_1\infty$] W. Hodges. From sentence meanings to full semantics. In [Gu$_0$vBPa$_5\infty$]. Forthcoming.

[Ho$_1$97a] W. Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL* 5(4):539–563, 1997.

[Ho$_1$97b] W. Hodges. Some strange quantifiers. In [MyRo$_4$Sa$_1$97, pp. 51–65].

[Ka$_4$83] M. Karpinski, ed. *Fundamentals of Computation Theory, Proceedings of the 1983 International FCT-Conference, Borgholm, Sweden, August 21–27, 1983*, *Lecture Notes in Computer Science* 158. Springer, 1983.

[MyRo$_4$Sa$_1$97] J. Mycielski, G. Rozenberg & A. Salomaa, eds. *Structures in Logic and Computer Science, A Selection of Essays in Honor of Andrzej Ehrenfeucht*, *Lecture Notes in Computer Science* 1261. Springer, 1997.

[Pa$_5$83] R. Parikh. Propositional logics of programs: new directions. In [Ka$_4$83, pp. 347–359].

[Se$_0$65] R. Selten. Spieltheoretische Behandlung eines Oligopolmodells mit Nachfrageträgheit. *Zeitschrift für die gesamte Staatwissenschaft* 121:301–324, 667–689, 1965.

[VäHo$_1\infty$] J. Väänänen & W. Hodges. Dependence of variables construed as an atomic formula. Preprint.

[vBtM96] J. van Benthem & A. ter Meulen, eds. *Handbook of Logic and Language*. Elsevier, 1996.

Reasoning about Communication Graphs

Eric Pacuit[1]
Rohit Parikh[2,3]

[1] Institute for Logic, Language and Computation
Universiteit van Amsterdam
1018 TV Amsterdam, The Netherlands

[2] Brooklyn College
City University of New York
2900 Bedford Avenue
Brooklyn, NY 11210, United States of America

[3] The Graduate Center
City University of New York
365 Fifth Avenue
New York City, NY 10016, United States of America

epacuit@science.uva.nl, rparikh@gc.cuny.edu

Abstract

Let us assume that some agents are connected by a *communication graph*. In the communication graph, an edge from agent i to agent j means that agent i can directly receive information from agent j. Agent i can then refine its own information by learning information that j has, including information acquired by j from another agent, k. We introduce a multi-agent modal logic with knowledge modalities and a modality representing communication among agents. Among other properties, we show that the logic is decidable, that it completely characterizes the communication graph, and that it satisfies the basic properties of the subset space logic of Moss and Parikh.

1 Introduction

The topic "who knew what and when" is not just of interest to epistemic logicians. Often it is the subject of political scandals (both real and imagined). For example, consider the much talked about Valerie Plame affair. A July 2003 column in the Washington Post reported that Plame was an undercover CIA operative. This column generated much controversy due to the fact that such information (the identity of CIA operatives) is restricted to the relevant government officials. Of course, in this situation, we know full well "Who knew what and when": in July of 2003, Robert Novak (the author of the article) knew that Plame was a CIA operative. What creates a scandal in this situation is *how* Novak came to know such information.

Since the CIA goes to great lengths to ensure that communication about sensitive information is contained within its own organization, the only way Novak could have known that Plame was a CIA operative was if a communication channel had been created between Novak and someone inside the CIA organization.

To put this a bit more formally, given a set of agents $\mathcal{A}$, call any graph $\mathcal{G} = (\mathcal{A}, E)$ a **communication graph** where the intended interpretation of an edge between agent i and agent j is that i and j *can communicate*. In this setting, the CIA can be represented as a connected component of $\mathcal{G}$. Given that the CIA is the only group of agents that (initially) knows the identity of CIA operatives, and Novak is not an element of the CIA component of $\mathcal{G}$ then we can conclude that Novak did not originally know the identity of CIA operatives and no amount of communication *that respects the graph* $\mathcal{G}$ can create a situation in which Novak does know the identity of a CIA operative. Thus Novak's report in the Washington Post implied that our original communication graph was incorrect[1]. That is, there must be an edge (or a chain) between Novak and *some* agent inside the CIA component. Since in principle, Novak could be connected to any member of the CIA component, much resources and time have been spent discussing the possible edges.

In this paper we develop[2] a multi-agent epistemic logic with a communication modality where agents are assumed to communicate according to some fixed communication graph. Agents are assumed to have some private information at the outset, but may refine their information by acquiring information possessed by other agents, possibly via yet other agents. That is, each agent is initially informed about the truth values of a finite set of propositional variables. Agents are assumed to be connected by a *communication graph*. In the communication graph, an edge from agent i to agent j means that agent i can directly receive information from agent j. Agent i can then refine its information by learning information that j has, including information acquired by j from another agent, k.

In keeping with the CIA-theme, we give an example from [Pa$_0$Pa$_5$05] of the type of situations that we have in mind. Let $\mathbf{K}_i\varphi$ mean that according to i's current information φ is true. Given a communication graph $\mathcal{G} = (\mathcal{A}, E)$, we say that a sequence of communications (i learns a fact from j who learns a fact from k, and so on) **respects the communication graph** if agents only communicate with their immediate neighbors in $\mathcal{G}$. Let $\Diamond\varphi$ mean that φ becomes true after a sequence of communications that respects

[1] Of course, it could also mean that we were incorrect about the agents' initial information — Novak could have had previous knowledge about the identity of CIA agents. In this paper, we are interested in studying communication and so will not consider this case.

[2] This framework was first presented in [Pa$_0$Pa$_5$05].

the communication graph. Suppose now that φ is a formula representing the exact whereabouts of Bin Laden, and that Bob, the CIA operative in charge of maintaining this information knows φ. In particular, $\mathbf{K}_{\text{Bob}}\varphi$, but suppose that at the moment, Bush does not know the exact whereabouts of Bin Laden ($\neg\mathbf{K}_{\text{Bush}}\varphi$). Presumably Bush *can* find out the exact whereabouts of Bin Laden ($\Diamond\mathbf{K}_{\text{Bush}}\varphi$) by going through CIA director Hayden, but of course, *we* cannot find out such information ($\neg\Diamond\mathbf{K}_{\text{E}}\varphi \wedge \neg\Diamond\mathbf{K}_{\text{R}}\varphi$) since we do not have the appropriate security clearance. Clearly, then, as a *prerequisite* for Bush learning φ, Hayden will also have to come to know φ. We can represent this situation by the following formula:

$$\neg\mathbf{K}_{\text{Bush}}\varphi \wedge \Box(\mathbf{K}_{\text{Bush}}\varphi \rightarrow \mathbf{K}_{\text{Hayden}}\varphi)$$

where $\Box$ is the dual of diamond ($\Box\varphi$ is true if φ is true after every sequence of communications that respect the communication graph).

Section 2 gives the details of our framework and the main results. Section 3 contains a discussion of some other relevant literature. We conclude the paper by discussing our underlying assumptions and provide pointers to future work.

2 The logic of communication graphs

This section describes the logic of communication graphs, $\mathcal{K}(\mathcal{G})$, introduced in [Pa$_0$Pa$_5$05]. The intended application is to reason about the flow of information among a group of agents whose communication is restricted by some communication graph. We begin by making some simplifying assumptions about the nature of the communication. Thus the logic presented here should be viewed as a first step towards a general logic to reason about the situations described in the Introduction.

Let $\mathcal{A}$ be a set of agents. A **communication graph** is a directed graph $\mathcal{G}_{\mathcal{A}} = (\mathcal{A}, E)$ where $E \subseteq \mathcal{A} \times \mathcal{A} - \{(i,i) \mid i \in \mathcal{A}\}$. Intuitively $(i,j) \in E$ means that i can directly receive information from agent j, but *without* j knowing this fact. Thus an edge from i to j in the communication graph represents a one-sided relationship between i and j. Agent i has access to *any* piece of information that agent j knows (but in a restricted language). We have introduced this 'one sidedness' restriction in order to simplify our semantics, but also because such situations of one sided learning occur naturally. A common situation that is helpful to keep in mind is accessing a website. We can think of agent j as creating a website in which everything he *currently* knows is available, and if there is an edge between i and j then agent i can access this website without j being aware that the site is being accessed. Another important application is *spying* where one person accesses another's information without the latter being aware that information is being leaked. Naturally j may have been able to access some other agent k's website and had updated some of her own information. Therefore, it is important to stress that when i accesses j's website, he is accessing j's current information which may include part of what k knew initially.

In order for any communication to take place, we must assume that the agents understand a common language. Thus we assume a set At of propositional variables, understood by all the agents, but with only specific agents knowing their actual values at the start. Letters p, q, etc., will denote elements of At. The agents will have some information – knowledge of the truth values of some elements of At, but refine that information by acquiring information possessed by other agents, possibly via yet other agents. This implies that if agents are restricted in whom they can communicate with, then this fact will restrict the knowledge they can acquire.

The assumption that i can access all of j's information is a significant idealization from common situations, but becomes more realistic if we think of this information as being confined to facts expressible as truth functional combinations of some small set of basic propositions. Thus our idealization rests on two assumptions:

1. All the agents share a common language, and
2. The agents make available all possible pieces of (purely propositional) information which they know *and which are expressible in this common language.*

The language is a multi-agent modal language with a communication modality. The formula $\mathbf{K}_i\varphi$ will be interpreted as "according to i's current information, i knows φ", and $\Diamond\varphi$ will be interpreted as "after some communications (which respect the communication graph), φ becomes true". For example the formula

$$\mathbf{K}_j\varphi \rightarrow \Diamond\mathbf{K}_i\varphi$$

is intended to express the statement: "If agent j (currently) knows φ, then after some communication, agent i can come to know φ". Let At be a finite set of propositional variables. A well-formed formula of $\mathcal{K}(\mathcal{G})$ has the following syntactic form

$$\varphi \ := p \mid \neg\psi \mid \varphi \wedge \psi \mid \mathbf{K}_i\varphi \mid \Diamond\varphi$$

where $p \in \mathsf{At}$. We abbreviate $\neg\mathbf{K}_i\neg\varphi$ and $\neg\Diamond\neg\varphi$ by $\mathbf{L}_i\varphi$ and $\Box\varphi$ respectively, and use the standard abbreviations for the propositional connectives ($\vee$, $\rightarrow$, and $\bot$). Let $\mathcal{L}_{\mathcal{K}(\mathcal{G})}$ denote the set of well-formed formulas of $\mathcal{K}(\mathcal{G})$. We also define $\mathcal{L}_0(\mathsf{At})$, (or simply $\mathcal{L}_0$ if At is fixed or understood), to be the set of ground formulas, i.e., the set of formulas constructed from At using $\neg, \wedge$ only.

2.1 Semantics

The semantics described here combines ideas both from the subset models of [Mo$_3$Pa$_5$92] and the history based models of Parikh and Ramanujam (see [Pa$_5$Ra$_2$85, Pa$_5$Ra$_2$03]). We assume that agents are initially given some private information and then communicate according to some fixed communication graph $\mathcal{G}$. The semantics in this section is intended to formalize what agents know and may come to know after some communication.

Initially, each agent i knows or is informed (say by nature) of the truth values of a certain subset At_i of propositional variables, and the At_i *as well as this fact are common knowledge.* Thus the other agents know that i knows the truth values of elements of At_i, but, typically, not what these values actually are. We shall not assume that the At_i are disjoint, but for this paper we *will* assume that the At_i together add up to all of At. Thus if At_i and At_j intersect, then agents i, j will share information at the very beginning. Let W be the set of boolean valuations on At. An element $v \in W$ is called a *state.* We use 1 for the truth value *true* and 0 for the truth value *false.* Initially each agent i is given a boolean valuation $v_i : \mathsf{At}_i \to \{0, 1\}$. This initial distribution of information among the agents can be represented by a vector $\vec{v} = (v_1, \dots, v_n)$. Of course, since we are modelling knowledge and not belief, these initial boolean valuations must be compatible. I.e., for each i, j, v_i and v_j agree on $\mathsf{At}_i \cap \mathsf{At}_j$. Call any vector of partial boolean valuations $\vec{v} = (v_1, \dots, v_n)$ **consistent** if for each $p \in \mathrm{dom}(v_i) \cap \mathrm{dom}(v_j)$, $v_i(p) = v_j(p)$ for all $i, j = 1, \dots, n$. Note that there is a 1-1 correspondence between consistent vectors and states w as we defined them earlier. We shall assume that only such consistent vectors arise as initial information. All this information is common knowledge and only the precise values of the v_i are private.

Definition 2.1. Let At be a finite set of propositional variables and $\mathcal{A} = \{1, \dots, n\}$ a finite set of agents. Given the distribution of sublanguages $\vec{\mathsf{At}} = (\mathsf{At}_1, \dots, \mathsf{At}_n)$, an **initial information vector for $\vec{\mathsf{At}}$** is any consistent vector $\vec{v} = (v_1, \dots, v_n)$ of partial boolean valuations such that for each $i \in \mathcal{A}$, $\mathrm{dom}(v_i) = \mathsf{At}_i$.

We assume that the only communications that take place are about the physical world. But we do allow agents to learn objective facts which are not atomic, but may be complex, like $p \vee q$ where $p, q \in \mathsf{At}$. Now note that if agent i learned some *literal* from agent j, then there is a simple way to update i's valuation v_i with this new information by just adding the truth value of another propositional symbol. However, if i learns a more general ground formula from agent j, then the situation will be more complex.

For instance if the agent knows p and learns $q \vee r$ then the agent now has three valuations on the set $\{p, q, r\}$ which cannot be described in terms of a partial valuation on a subset of At.

Fix a communication graph $\mathcal{G}$ and suppose that agent i learns some ground fact φ (directly) from agent j. Of course, there must be an edge from agent i to agent j in $\mathcal{G}$. This situation will be represented by the tuple (i, j, φ) and will be called a **communication event**. For technical reasons we assume that all formulas in a communication event are expressed in a canonical *disjunctive normal form* (DNF). That is, we assume φ is a set $\{C_1, \ldots, C_k\}$ where each C_i is a consistent finite set of elements of At and their negations. Let $\mathcal{L}_{\mathrm{DNF}}(\mathsf{At})$ be the set of all such sets. Each set $\{C_1, \ldots, C_k\}$ represents the formula $\bigvee_{i=1,\ldots,k} \bigwedge C_i$. Recall that for each formula $\psi \in \mathcal{L}_0(\mathsf{At})$ there is a unique element $\{C_1, \ldots, C_k\} \in \mathcal{L}_{\mathrm{DNF}}(\mathsf{At})$ such that $\bigvee_{i=1,\ldots,k} \bigwedge C_i$ is logically equivalent to ψ. In what follows, we shall sometimes use φ to mean either a formula (an element of $\mathcal{L}(\mathsf{At})$ or $\mathcal{L}_0(\mathsf{At})$) or an element of $\mathcal{L}_{\mathrm{DNF}}(\mathsf{At})$ and trust that this ambiguity of notation will not cause any confusion. Of course we could use a unique element of $\mathcal{L}_0(\mathsf{At})$ to express a member of $\mathcal{L}_{\mathrm{DNF}}(\mathsf{At})$ and that too would work.

Definition 2.2. Let $\mathcal{G} = (\mathcal{A}, E_{\mathcal{G}})$ be a communication graph. A tuple (i, j, φ), where $\varphi \in \mathcal{L}_{\mathrm{DNF}}(\mathsf{At})$ and $(i, j) \in E_{\mathcal{G}}$, is called a **communication event**. Then $\Sigma_{\mathcal{G}} = \{(i, j, \varphi) \mid \varphi \in \mathcal{L}_{\mathrm{DNF}}, (i, j) \in E_{\mathcal{G}}\}$ is the set of all possible communication events (given the communication graph $\mathcal{G}$).

The following fact will be needed in what follows. The proof is well-known and is left to the reader.

Lemma 2.3. Suppose that there are k elements in At and n elements in $\mathcal{A}$. Then for any communication graph $\mathcal{G}$, there are at most $n \times n \times (2^{2^k})$ elements in $\Sigma_{\mathcal{G}}$.

Given the set of events $\Sigma_{\mathcal{G}}$, a **history** is a finite sequence of events. I.e., $H \in \Sigma_{\mathcal{G}}^*$. The empty history will be denoted ε. The following notions are standard (see [Pa$_5$Ra$_2$85, Pa$_5$Ra$_2$03] for more information). Given two histories H, H', say $H \preceq H'$ if and only if $H' = HH''$ for some history H'', i.e., H is an initial segment of H'. Obviously, $\preceq$ is a partial order. If H is a history, and (i, j, φ) is a communication event, then H followed by (i, j, φ) will be written $H; (i, j, \varphi)$. Given a history H, let $\lambda_i(H)$ be i's local history corresponding to H. I.e., $\lambda_i(H)$ is a sequence of events that i can "see". Given our assumption of "one-sided communication", for this paper we use the following definition of λ_i: Map each event of the form (i, j, φ) to itself, and map other events (m, j, ψ) with $m \neq i$ to the null string while preserving the order among events.

Definition 2.4. Fix a finite set of agents $\mathcal{A} = \{1, \ldots, n\}$ and a finite set of propositional variables At along with subsets $(\mathsf{At}_1, \ldots, \mathsf{At}_n)$. A **communication graph frame** is a pair $\langle \mathcal{G}, \vec{\mathsf{At}} \rangle$ where $\mathcal{G}$ is a communication graph, and $\vec{\mathsf{At}} = (\mathsf{At}_1, \ldots, \mathsf{At}_n)$ is an assignment of sub-languages to the agents. A **communication graph model** based on a frame $\langle \mathcal{G}, \vec{\mathsf{At}} \rangle$ is a triple $\langle \mathcal{G}, \vec{\mathsf{At}}, \vec{v} \rangle$, where $\vec{v}$ is an initial information vector for $\vec{\mathsf{At}}$.

Now we address two issues. One is that not all histories are legal. For an event (i, j, φ) to take place after a history H, it must be the case that $(i, j) \in E_{\mathcal{G}}$, and that after H (and before (i, j, φ)), j already knew φ. Clearly i cannot learn from j something which j did not know. Whether a history is justified depends not only on the initial valuation, but also on the set of communications that have taken place prior to each communication in the history.

The second issue is that the information which an agent learns by "reading" a formula φ may be *more* than just the fact that φ is true. For suppose that i learns $p \vee q$ from j, but j is not connected, directly or indirectly, to anyone who might know the initial truth value of q. In this case i has learned *more* than $p \vee q$, i has learned p as well. For the only way that j could have known $p \vee q$ is if j knew p in which case p must be true. Our definition of the semantics below will address both these issues.

We first introduce the notion of i-equivalence among histories. Intuitively, two histories are i-equivalent if those communications which i takes active part in, are the same.

Definition 2.5. Let w be a state and H a finite history. Define the relation $\sim_i$ as follows: $(w, H) \sim_i (v, H')$ if and only if $w{\restriction}\mathsf{At}_i = v{\restriction}\mathsf{At}_i$ and $\lambda_i(H) = \lambda_i(H')$.

Formulas will be interpreted at pairs (w, H) where w is a state (boolean valuation) and H is a history (a finite sequence of communication events).

To deal with the notion of legal or justified history we introduce a propositional symbol L which is satisfied only by legal pairs (w, H). (We may also write $L(w, H)$ to indicate that the pair (w, H) is legal.) Since L can only be defined in terms of knowledge, and knowledge in turn requires quantification over legal histories, we shall need mutual recursion.

Definition 2.6. Given a communication graph and the corresponding model $\mathcal{M} = \langle \mathcal{G}, \vec{\mathsf{At}}, \vec{v} \rangle$, and pair (w, H), we define the legality of (w, H) and the truth $\models_{\mathcal{M}}$ of a formula as follows:

- $w, \varepsilon \models_{\mathcal{M}} L$
- $w, H; (i, j, \varphi) \models_{\mathcal{M}} L$ iff $w, H \models_{\mathcal{M}} L$, $(i, j) \in E$ and $w, H \models_{\mathcal{M}} \mathbf{K}_j \varphi$
- $w, H \models_{\mathcal{M}} p$ iff $w(p) = 1$, where $p \in \mathsf{At}$
- $w, H \models_{\mathcal{M}} \neg\varphi$ iff $w, H \not\models_{\mathcal{M}} \varphi$

- $w, H \models_{\mathcal{M}} \varphi \wedge \psi$ iff $w, H \models_{\mathcal{M}} \varphi$ and $w, H \models_{\mathcal{M}} \psi$
- $w, H \models_{\mathcal{M}} \Diamond\varphi$ iff $\exists H',\ H \preceq H',\ L(w, H'),$ and $w, H' \models_{\mathcal{M}} \varphi$
- $w, H \models_{\mathcal{M}} \mathbf{K}_i\varphi$ iff $\forall (v, H')$ if $(w, H) \sim_i (v, H')$, and $L(v, H')$, then $v, H' \models_{\mathcal{M}} \varphi$

Unless otherwise stated, we shall only consider legal pairs (w, H), i.e., pairs (w, H) such that $w, H \models L$. We say φ is **valid in $\mathcal{M}$**, $\models_{\mathcal{M}} \varphi$ if for all (w, H), $w, H \models_{\mathcal{M}} \varphi$. Finally, we say φ is **valid in the communication graph frame** $\mathcal{F}$ if φ is valid in all models based on $\mathcal{F}$.

There are two notions of validity relevant for our study. The first is relative to a fixed communication graph. Let $\mathcal{G}$ be a fixed communication graph. We say that a formula $\varphi \in \mathcal{L}_{\mathcal{K}(\mathcal{G})}$ is **$\mathcal{G}$-valid** provided φ is valid in all communication graph frames $\mathcal{F}$ based on $\mathcal{G}$. A formula $\varphi \in \mathcal{L}_{\mathcal{K}(\mathcal{G})}$ is **valid** if φ is $\mathcal{G}$-valid for all communication graphs $\mathcal{G}$. Of course validity implies $\mathcal{G}$-validity, but not vice versa. We write $\models_{\mathcal{G}} \varphi$ if φ is $\mathcal{G}$-valid and $\models \varphi$ if φ is valid.

Some comments are in order concerning the above definition of truth. L is defined in terms of the knowledge operator, and the knowledge operator uses L in its definition. This definition is of course fine since the definition uses recursion on the length of the formula. Still, it is not clear that the process of determining whether a history is legal will terminate in a *finite* amount of time. For whether or not the history-state pair $(w, H; (i, j, \varphi))$ is legal depends on whether (w, H) satisfies $\mathbf{K}_j\varphi$ which, in turn, depends on a set of histories which may be *longer* than H. We now show that the process of determining whether a history is legal will terminate.

We first need some notation. A **one-step compression** of H is a history H' which is obtained by deleting *one* second or subsequent occurrences of an event (i, j, φ) from H. I.e., if (i, j, φ) has occurred twice, then eliminate some later occurrence. Let $\mathrm{c}(H)$ denote the maximally compressed history. The history $\mathrm{c}(H)$ is generated by including each (i, j, φ) event of H exactly once according to the following order on events: e comes before e' iff the first occurrence of e in H came before the first occurrence of e' in H. The key observation is that the legality of a history-state pair (w, H) depends only on the legality of the pair $(w, \mathrm{c}(H))$.

Lemma 2.7. Let $\mathcal{G}$ be a communication graph, $\Sigma_{\mathcal{G}}$ a set of events and H be any history over $\Sigma_{\mathcal{G}}$. Suppose that w is a state. Then

- For all φ, and all j, $w, H \models \mathbf{K}_j\varphi$ iff $w, \mathrm{c}(H) \models \mathbf{K}_j\varphi$
- (w, H) is legal iff $(w, \mathrm{c}(H))$ is legal.

Proof. Clearly it is sufficient to prove the two conditions when $\mathrm{c}(H)$ is replaced by an H' obtained from H by the elimination of *one* extra event. Therefore we shall make this assumption in the rest of the proof. Thus $H = H_1eH_2eH_3$ and $H' = H_1eH_2H_3$. Here e is some event (i, j, φ).

We show that $(w, H), (w, H')$ satisfy the same formulas ψ. Clearly this is true if ψ is atomic and the argument also goes through for boolean combinations.

Suppose $\psi = \Diamond\theta$ and $w, H \models \psi$. Then there is H_4 such that $w, H; H_4 \models \theta$. By induction hypothesis $w, H'; H_4 \models \theta$ (it is not hard to see that it is legal) and hence $w, H' \models \Diamond\psi$. The converse is similar.

Suppose $\psi = \mathbf{K}_r\theta$ and $w, H \models \psi$ where $r \neq i$. This case is easy as H, H' are r-equivalent.

What if we have $r = i$? $w, H \models \mathbf{K}_i\theta$ iff for all v, H'' such that $(v, H'') \sim_i (w, H)$, $v, H'' \models \theta$. But since e has already occurred in both H, H', the possible v in question are the same for both. The H'' will have two e events and eliminating the second e will yield an H''' such that $v, H''' \models \theta$ iff $w, H'' \models \theta$. Thus the case $r = i$ also works.

The proof that compression preserves legality or illegality is now immediate for it depends on some knowledge formulas being true. But that issue is not affected by the elimination of extra events e. Q.E.D.

With Lemma 2.7 we can show that the process of determining if a state-history pair (w, H) is legal terminates. More formally,

Proposition 2.8. Let $\mathcal{M} = \langle \mathcal{G}, \vec{\mathsf{At}}, \vec{v} \rangle$ be a communication graph model. For any $H \in \Sigma_{\mathcal{G}}^*$ and state w, the question "Does $(w, H) \models L$?" terminates in a finite amount of time.

Proof. This is now immediate by the previous lemma. When asking whether $w, H \models \mathbf{K}_i\varphi$ we need to look at pairs (v, H') which are i-equivalent to w, H. But now we can confine ourselves to H' is which no (r, j, ψ) event with $r \neq i$ occurs twice, and these are bounded in length. Q.E.D.

2.2 Some results

We now state the basic results about the logic of communication graphs.

We already defined $\mathrm{c}(H)$ earlier. We say that two histories are ***c*-equivalent**, written $C(H, H')$, if $\mathrm{c}(H) = \mathrm{c}(H')$. Clearly H, H' have the same semantic properties as $\mathrm{c}(H) = \mathrm{c}(H')$ and hence as each other. Moreover, one is legal iff the other is legal. It follows also that for every H_1, $H; H_1$ is legal iff $H'; H_1$ is. Thus C is a bisimulation.

In the following, a history H is called ***w*-maximal** if (w, H) is legal and all possible (finitely many) communication events have taken place at least once. An interesting consequence of the above lemma is the existence of a **maximal history** (relative to some w).

Theorem 2.9.

1. If a formula φ is satisfiable in some graph model $(\mathcal{G}, \vec{\mathsf{At}})$ then it is satisfiable in a history in which no communication (i, j, φ) occurs twice.
2. If H is w-maximal, then for all formulas $\varphi \in \mathcal{L}_{\mathcal{K}(\mathcal{G})}$, $\models \varphi \to \Box\varphi$.
3. If H is w-maximal and H' is any history compatible with w such that $H \preceq H'$, then for all formulas $\varphi \in \mathcal{L}_{\mathcal{K}(\mathcal{G})}$, if $w, H' \models \varphi$ then $w, H \models \varphi$.
4. If H and H' are w-maximal, then for each formula $\varphi \in \mathcal{L}_{\mathcal{K}(\mathcal{G})}$, $w, H \models \varphi$ iff $w, H' \models \varphi$.

Proof. The first three parts follow from Lemma 2.7. We shall prove the last statement: for any state w, if H and H' are w-maximal histories, then (w, H) and (w, H') satisfy the same formulas. The proof is by induction on φ.

The base case and boolean connectives are straightforward. Suppose that φ is of the form $\Diamond\psi$. Let w be an arbitrary state and suppose that H and H' are w-maximal histories. Suppose that $(w, H) \models \Diamond\psi$. Then there is some H'' such that $H \preceq H''$, (w, H'') is legal and $(w, H'') \models \varphi$. By part 3 above, $w, H \models \psi$ and by the induction hypothesis, $w, H' \models \psi$. Hence, $w, H' \models \Diamond\psi$. Thus if $w, H \models \Diamond\psi$ then $w, H' \models \Diamond\psi$. The other direction is similar.

For the knowledge case we need the following claim:

Claim 2.10. Let w be a state. Suppose H_1 and H_2 are w-maximal and (v, H_3) is legal. If $(w, H_1) \sim_i (v, H_3)$, then there is a history H_4 which is v-maximal such that $(w, H_2) \sim_i (v, H_4)$.

Proof of Claim. Let w be a state and suppose that H_1 and H_2 are w-maximal histories and (v, H_3) is a legal history-state pair such that $(w, H_1) \sim_i (v, H_3)$. Then v and w must agree on every atom which is not only known to i, but also on *every* atom known to some other agent whom i can read directly or indirectly. For at maximality, i already knows the truth values of all the sets At_j where j is directly or indirectly accessible from i. Thus we can find a legal history-state pair (v, H_4') such that $(w, H_2) \sim_i (v, H_4')$. It is not hard to see that H_4' can be extended to a v-maximal history. Q.E.D. (Claim 2.10)

Returning to the induction proof. Suppose that φ is of the form $\mathbf{K}_i\psi$, w is an arbitrary state, and H and H' are w-maximal histories. Suppose that $(w, H) \models \mathbf{K}_i\psi$ and $(w, H') \not\models \mathbf{K}_i\psi$. Then there is a history state pair (v, H'') such that $(w, H') \sim_i (v, H'')$, (v, H'') is legal and $v, H'' \not\models \psi$. Note that without loss of generality we can assume that H'' is v-maximal (every legal history-state pair can be extended to a maximal history, furthermore this extension cannot differ on the truth value of φ since H'' already contains *all* events of the form (i, j, χ)). By the above claim (let $H_1 = H'$, $H_2 = H$ and $H_3 = H''$), there is a v-maximal history H''' with $(w, H) \sim_i (v, H''')$. By the induction hypothesis, (v, H'') and (v, H''') satisfy the same formulas. Hence $(w, H) \sim_i (v, H''')$ and $v, H''' \not\models \psi$. This contradicts the assumption that $w, H \models \mathbf{K}_i\psi$. Hence, for any w-maximal histories H and H', $w, H \models \mathbf{K}_i\psi$ iff $w, H' \models \mathbf{K}_i\psi$. Q.E.D.

This result immediately gives us a decision procedure as we can limit the length of the history which might satisfy some given formula φ.

Notice that if we restrict our attention to maximal histories, then the following property will be satisfied: for any two agents i and j, if there is a path in the communication graph from i to j, then any ground fact that j knows, i will also know. In this case, we can say that j *dominates* i. This is Fitting's "dominance" relation discussed in Section 3.

The following simple result demonstrates that given any sequence of communications H, the agents know at least the set of formulas that are implied by the set of formulas in H. That is, given a legal pair (w, H), let $X_i(w, H)$ be the set of states that agent i considers possible if the actual state is w and the communication between the agents evolved according to H. Given a formula $\varphi \in \mathcal{L}_0(\mathsf{At})$, let $\widehat{\varphi} = \{w \mid w \in W,\ w(\varphi) = 1\}$. Now we formally define $X_i(w, H)$ recursively as follows

1. $X_i(w, \varepsilon) = \{v \mid v{\restriction}\mathsf{At}_i = w{\restriction}\mathsf{At}_i\}$
2. $X_i(w, H; (i, j, \varphi)) = X_i(w, H) \cap \hat{\varphi}$
3. if $i \neq m$ then $X_i(w, H; (m, j, \varphi)) = X_i(w, H)$

The following theorem shows that agents know at least the formulas implied by the set $X_i(w, H)$. The proof can be found in [Pa$_0$Pa$_5$05] and will not be repeated here.

Theorem 2.11. Let $\mathcal{M} = \langle \mathcal{G}, \vec{\mathsf{At}}, \vec{v} \rangle$ be any communication graph model and φ a ground formula. If $X_i(w, H) \subseteq \widehat{\varphi}$, then $(w, H) \models_{\mathcal{M}} \mathbf{K}_i(\varphi)$.

As we saw above, the converse is not true. That is, there are formulas that an agent can come to know which are not implied by the set $X_i(w, H)$. These are the formulas that agents can deduce given their knowledge of the communication graph.

The following axioms and rules are known to be sound and complete with respect to the family of all subset spaces [Mo$_3$Pa$_5$92]. Thus they represent the core set of axioms and rules for any **topologic** (see [Pa$_5$Mo$_3$St$_4$07] for the current state of affairs).

1. All propositional tautologies
2. $(p \to \Box p) \wedge (\neg p \to \Box \neg p)$, for $p \in \mathsf{At}$
3. $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$
4. $\Box\varphi \to \varphi$
5. $\Box\varphi \to \Box\Box\varphi$
6. $\mathbf{K}_i(\varphi \to \psi) \to (\mathbf{K}_i\varphi \to \mathbf{K}_i\psi)$
7. $\mathbf{K}_i\varphi \to \varphi$
8. $\mathbf{K}_i\varphi \to \mathbf{K}_i\mathbf{K}_i\varphi$
9. $\neg\mathbf{K}_i\varphi \to \mathbf{K}_i\neg\mathbf{K}_i\varphi$
10. (Cross axiom) $\mathbf{K}_i\Box\varphi \to \Box\mathbf{K}_i\varphi$

We include the following rules: modus ponens, $\mathbf{K}_i$ necessitation and $\Box$ necessitation. It is easy to verify that the above axioms and rules are valid, i.e., valid in all frames based on any communication graphs. We only demonstrate that the cross axiom $\mathbf{K}_i\Box\varphi \to \Box\mathbf{K}_i\varphi$ is valid. It is easier to consider it in its contrapositive form: $\Diamond\mathbf{L}_i\varphi \to \mathbf{L}_i\Diamond\varphi$. This is interpreted as follows: if there is a sequence of updates that lead agent i to consider φ possible, then i already thinks it possible that there is a sequence of updates after which φ becomes true.

Proposition 2.12. The axiom scheme $\Diamond\mathbf{L}_i\varphi \to \mathbf{L}_i\Diamond\varphi$ is valid.

Proof. Let $\mathcal{G}$ be an arbitrary communication graph and $\mathcal{M} = \langle \mathcal{G}, \vec{\mathsf{At}}, \vec{v} \rangle$ any communication graph model based on $\mathcal{G}$. Suppose that (w, H) is a legal state-history pair. Suppose that $w, H \models \Diamond\mathbf{L}_i\varphi$. Then there exists H' with $H \preceq H'$ such that $w, H' \models \mathbf{L}_i\varphi$. Hence there is a pair (v, H'') such that $(w, H') \sim_i (v, H'')$ and $v, H'' \models_{\mathcal{M}} \varphi$. Let H''' be any sequence such that $\lambda_i(H) = \lambda_i(H''')$ and $H''' \preceq H''$. Such a history must exist since $H \preceq H'$ and $H' \sim_i H''$. Since $H \preceq H'$, $\lambda_i(H) \preceq \lambda_i(H') = \lambda_i(H'')$. Therefore, we need only let H''' be any initial segment of H'' containing $\lambda_i(H)$. By definition of L, all initial sequences of a legal history are legal. Therefore, since $v, H'' \models_{\mathcal{M}} \varphi$, $v, H''' \models \Diamond\varphi$; and since $H \sim_i H'''$, $w, H \models_{\mathcal{M}} \mathbf{L}_i\Diamond\varphi$. Q.E.D.

The next lemma follows from the existence of maximal histories.

Lemma 2.13. The axiom $\Box\Diamond\varphi \leftrightarrow \Diamond\Box\varphi$ is valid.

Proof. Let $\mathcal{G}$ be an arbitrary communication graph, $\mathcal{M} = \langle \mathcal{G}, \vec{\mathsf{At}}, \vec{v} \rangle$ any communication graph model based on $\mathcal{G}$ and (w, H) a legal history-state pair. Suppose that $w, H \models_{\mathcal{M}} \Box\Diamond\varphi$. Let H' be a w-maximal history extending H, then $w, H' \models_{\mathcal{M}} \Diamond\varphi$ and hence there is a history H'' such that $H' \preceq H''$ and $w, H'' \models_{\mathcal{M}} \varphi$. Since H' is maximal, by Theorem 2.9 $w, H' \models \varphi$. By Theorem 2.9 again, $w, H' \models \Box\varphi$. Hence $w, H \models \Diamond\Box\varphi$.

Conversely, suppose that $w, H \models_{\mathcal{M}} \Diamond\Box\varphi$. Then there is a history H' such that $H \preceq H'$ and $w, H' \models_{\mathcal{M}} \Box\varphi$. Let H'_m be a w-maximal history that extends H'. Then $w, H'_m \models_{\mathcal{M}} \varphi$. Let H'' be any history that extends H and H''_m be a w-maximal history that extends H''. By Theorem 2.9, since $w, H'_m \models_{\mathcal{M}} \varphi$, $w, H''_m \models_{\mathcal{M}} \varphi$. Hence $w, H'' \models_{\mathcal{M}} \Diamond\varphi$ and hence $w, H \models_{\mathcal{M}} \Box\Diamond\varphi$. Q.E.D.

If we fix a communication graph, then there are more formulas which are valid.

Lemma 2.14. Let $\mathcal{G} = (\mathcal{A}, E)$ be a communication graph. Then for each $(i, j) \in E$, for all $\ell \in \mathcal{A}$ such that $\ell \neq i$ and $\ell \neq j$ and all ground formulas φ, the scheme

$$\mathbf{K}_j\varphi \wedge \neg\mathbf{K}_\ell\varphi \to \Diamond(\mathbf{K}_i\varphi \wedge \neg\mathbf{K}_\ell\varphi)$$

is $\mathcal{G}$-valid.

Proof. Let $\mathcal{G}$ be an arbitrary communication graph and $\mathcal{M}$ a communication graph model based on $\mathcal{G}$. Suppose that $w, H \models_{\mathcal{M}} \mathbf{K}_j\varphi \wedge \neg\mathbf{K}_\ell\varphi$. Then j knows φ and hence i can read φ directly from j's website. More formally, $H; (i, j, \varphi)$ is a legal history (provided that H is legal). The agent ℓ is none the wiser as $\lambda_\ell(H) = \lambda_\ell(H; (i, j, \varphi))$. Therefore, $w, H; (i, j, \varphi) \models_{\mathcal{M}} \mathbf{K}_i\varphi \wedge \neg\mathbf{K}_\ell\varphi$. Q.E.D.

The converse is not quite true. For suppose that agent i is connected to agent j via (exactly) two other agents ℓ_1, ℓ_2. Then a fact known to j can be learned by i without ℓ_1 finding out about it, ditto for ℓ_2 and for any agent beside ℓ_1, ℓ_2. However, in this scenario, it is impossible for i to find out some φ from j unless ℓ_1 and ℓ_2 jointly know φ.

3 Related literature

Communication graphs, or communication networks, and more generally social networks[3] have been studied by a number of different communities. Most notably, social networks are an important topic in sociology. But computer scientists have also had quite a lot to say about networks. The

[3] A social network is a graph on a set of agents where edges represent some social interaction such as acquaintances, coauthors on a mathematical paper, costars in a movie, and so on.

goal of this section is not to survey this vast amount of literature, but rather to give some details about a few papers most relevant to the framework discussed in Sections 1 and 2. Needless to say, this list of topics is not meant to be complete.

Coordination problems

Suppose there are two generals A and B with armies at the top of two mountains with a valley in between them. As the story goes, the generals attempt to coordinate their action by sending messages back and forth. However, since the communication channel is unreliable (the messengers must travel through dangerous territory), common knowledge of the time to attack cannot be achieved. This puzzle, called the generals problem, has been topic of much discussion. See [Fa+95] for a formal treatment and discussion of relevant literature and [Le$_2$69] for a discussion of common knowledge as it relates to coordination problems. Of course, if there was a reliable communication channel between the two generals, then they could easily coordinate their actions. Thus the existence of a communication graph (in this case just an edge between the two generals) facilitates coordination. This raises some interesting questions about the structure of the communication graph on a group of agents and its affect on coordination.

In [Ch$_4$00, Ch$_4$99], Michael Chwe investigates the general question of when the structure of the communication graph can facilitate coordination among a group of agents. Chwe considers the following situation. There is a finite set of agents $\mathcal{A}$. Each agent must decide whether or not to revolt against the current government. That is, assume that agents choose between r (revolt) and s (stay at home). The agents are assumed to use the following decision rule: "I'll go if you go". That is, for a particular agent i, the greater the number of agents i believes will choose r, the higher the utility i assigns to r, provided agent i is willing to revolt. More formally, it is assumed that each agent is either willing to revolt (w) or not willing to revolt (nw). Then a utility function for agent i maps elements of $\{w, nw\} \times \{r, s\}^n$ to real numbers with the constraint that if an agent is not willing to revolt, then the utility of revolting is zero regardless the opinions of i's neighbors. It is assumed that the agents are connected by a communication graph $\mathcal{G} = (\mathcal{A}, E)$. Here $(i, j) \in E$ is intended to mean "agent i talks to agent j". That is i informs j as to whether or not he will revolt. Thus the set $B_i = \{j \mid (j, i) \in E\}$ is the set of agents for which i knows which action they will perform. Finally, it is assumed that the communication graph is common knowledge and that each agent has a prior belief[4] about which agents will revolt.

[4] Beliefs are represented by probability functions over the set $\{r, s\}^n$.

Chwe considers the question "which communication graphs enable the group to revolt?" To that end, a strategic game $\Gamma(\mathcal{G}, \{\pi\}_{i\in\mathcal{A}})$ is defined, where π_i is agent i's prior beliefs. In this game, an agent i's decision to revolt depends on its prior beliefs π_i and the set B_i of agents that i has communicated with. Of course if agent i's prior belief assigns high enough probability to a large enough group of agents revolting, then that agent will revolt *regardless of the communication graph.* Thus the interesting question is which communication graph will enable the group to revolt *regardless of the agents' prior probabilities.* Chwe calls such communication graphs a **sufficient network**.

The main result of the paper [Ch$_4$00] is a characterization of minimal sufficient networks. First of all, it should be obvious that if a communication graphs $\mathcal{G}$ enables a group to revolt, then so will any communication graph $\mathcal{G}'$ which is just like $\mathcal{G}$ except with additional edges (it is straightforward to prove this in Chwe's framework). Chwe showed that any sufficient network has the following property: there is a finite set of cliques such that

1. Each agent is in at least one clique,
2. there is a relation $\to$ between the cliques that characterizes the edge relation in the graph. That is there is an edge from i to j iff there is some clique containing i and some clique containing j that are related by $\to$, and
3. the cliques are totally ordered by $\to$.

This result provides an interesting perspective on collective action. The communication graph facilitates the group's ability to share information and thus enabling group action. The logic presented in Section 2 is intended to make clear precisely how an agent's information can change in situations similar to the one described above.

Agreeing to disagree

In 1976, Aumann proved a fascinating result [Au76]. Suppose that two agents have the same prior probability and update their probabilities of an event E with some private information using Bayes' rule. Then Aumann showed that if the posterior probability of E is common knowledge, then they must assign the *same* posterior probability to the event E. In other words, if agents have the same prior probability and update using Bayes' rule, then the agents cannot "agree to disagree" about their posterior probabilities. See [Bo$_0$Ne97] for a nice discussion of this result and the literature that it generated. An immediate question that comes to mind is "*How* do the posterior probabilities become common knowledge?" Starting with Geanakoplos and Polemarchakis [$\text{Ge}_0\text{Po}_1$82], a number of papers have addressed this issue [$\text{Ca}_5$83, $\text{Ba}_1$85, $\text{Pa}_5\text{Kr}_0$90, $\text{Kr}_0$96, $\text{He}_0$96].

The key idea is that common knowledge arises through communication. Suppose there are two agents who agree on a prior probability function. Suppose that each agent receives some private information concerning an event E and updates their probability function accordingly. Geanakoplos and Polemarchakis [$\text{Ge}_0\text{Po}_1$82] show that if the agents each announce their posterior probabilities and update with this new information, then the probabilities will eventually become common knowledge and the probabilities will be equal. Similar to Chwe's analysis described above, the existence of a communication graph (with an edge between the two agents) enables consensus about the posterior probabilities.

Parikh and Krasucki [$\text{Pa}_5\text{Kr}_0$90] look at the general situation where there may be more than two agents[5] and communication is restricted by a communication graph. They show that under certain assumptions about the communication graph, consensus can be reached even though the posterior probabilities of the agents may not be common knowledge[6]. Before stating their result, some clarification is needed. Note that a communication graph tells us which agent *can* communicate with which agent, but not when two agents *do* communicate. To represent this information, Parikh and Krasucki introduce the notion of a **protocol**. A protocol is a pair of functions (r, s) where $r : \mathbb{N} \to \mathcal{A}$ and $s : \mathbb{N} \to \mathcal{A}$. Intuitively, $(r(t), s(t))$ means that $r(t)$ receives a message from $s(t)$ at time t. Say a protocol (r, s) respects a communication graph $\mathcal{G} = (\mathcal{A}, E)$ if for each $t \in \mathbb{N}$, $(r(t), s(t)) \in E$. A protocol is said to be **fair** provided every agent can send a message to any other agent, either directly or indirectly, infinitely often[7]. Parikh and Krasucki show that if the agents are assumed to have finitely many information sets, then for any protocol if the agents sends the current probability[8] (conditioned on the agent's current information set) of proposition A, then after a finite amount of time t for each agent i, the messages received after time t will not change i's information set. Furthermore, if the protocol is assumed

[5] Cave [$\text{Ca}_5$83] also considers more than two agents, but assumes all communications are public announcements.

[6] This point was formally clarified by Heifetz in [$\text{He}_0$96]. He demonstrates how to enrich the underlying partition space with time stamps in order to formalize precisely when events become common knowledge.

[7] Consult [$\text{Pa}_5\text{Kr}_0$90] for a formal definition of "fairness".

[8] Actually, Parikh and Krasucki consider a more general setting. They assume agents communicate the value of some function f that maps events to real numbers. Intuitively, f could be thought of as the expected value of a random variable given some information set. The only condition imposed on f is a convexity condition: for any two disjoint close subsets X and Y, $f(X \cup Y)$ lies in the open interval between $f(X)$ and $f(Y)$. Here *closed* is defined with respect to the agents' information sets. This generalizes a condition imposed by Cave [$\text{Ca}_5$83] and Bacharach [$\text{Ba}_1$85].

to be fair (i.e., the communication graph is strongly connected) then all the agents will eventually assign the same probability to A. Krasucki takes the analysis further in [$\mathrm{Kr}_0$96] and provides conditions on the protocol (and implicitly on the underlying communication graph) which will guarantee consensus regardless of the agents' initial information.

Similar to Chwe's analysis, Parikh and Krasucki's analysis shows that the structure of the communication graph is a key aspect of consensus in a group. We end this section with a different interpretation of a communication graph that provides a very interesting perspective on many-valued modal logic.

Many-valued modal logic

In [$\mathrm{Fi}_2$91a, $\mathrm{Fi}_2$91b], Fitting discusses various families of many-valued modal logics. In these frameworks, Fitting assumes that both the valuation of the formulas and the accessibility relation are many-valued. The frameworks are motivated in [$\mathrm{Fi}_2$91b] with the help of a structure similar to a communication graph. As in this paper, it is assumed that there is a graph with the set of agents as nodes. However, Fitting interprets an edge in this graph differently: "i is related to j" means that i *dominates* j, where 'dominate' means that "j says that something is true whenever i says it is". It is assumed that this relation is a partial order. Each agent is assumed to have its own valuation and accessibility relation on a set of possible worlds. Fitting is interested in which modal formulas the agents will agree on in a particular world. Two semantics are presented that solve this problem.

Deciding which agents agree on a particular formula in a common language naturally suggests a many-valued modal logic where formulas are assigned subsets of agents (i.e., the subsets of the agents are the set of *truth values* in this many-valued setting). Suppose that φ is assigned the set $\mathcal{B} \subseteq \mathcal{A}$ in a state w, then the intended interpretation is that each agent in $\mathcal{B}$ agrees on the truth value of φ in w. The domination relation outlaws certain subsets of agents as truth values. For example, if i dominates j, then $\{i\}$ cannot be a possible truth value since j is assumed to *always* agree with i on the set of true formulas. The domination relation also provides an intuitionistic flavor to the underlying logic. For example, consider the formula $\neg\varphi$ and suppose i dominates j. Now it is consistent with the notion of domination that i can consider φ false and j considers φ true. In this case, if we interpret $\neg$ classically, then i considers $\neg\varphi$ true while j considers $\neg\varphi$ false, which contradicts the fact that i dominates j. Thus, we are forced to say that i considers $\neg\varphi$ true if *i and all agents that i dominates* consider that φ is false. Fitting offers two semantics which take these observations into account. One is a combination of Kripke intuitionistic models and Kripke multi-modal models and the second is a many-valued Kripke modal model. The two semantics are shown to be equivalent and a sound and complete axiomatization is offered.

Rasiowa and Marek offer a similar interpretation of a communication graph [$\text{Ra}_5\text{Ma}_5$89, $\text{Ra}_5\text{Ma}_5$93]. In their framework an edge from i to j means that "j is more perceptive than i". If this is the case, then j's valuation of a proposition p is "better than" i's valuation of that same variable. Rasiowa and Marek provide a framework to reason about formulas on which there is consensus among the agents. The framework discussed in the rest of this paper can be seen as an attempt to explain *how* an agent i can come to dominate another agent j. That is, assuming the agents start with consistent (partial) theories, i can dominate j if there is a (possibly indirect) communication channel from i to j and j asks i about the truth value of all formulas in the language.

Dynamic epistemic semantics

The study of *Dynamic Epistemic Logic* attempts to combine ideas from dynamic logics of actions and epistemic logic. The main idea is to start with a formal model that represents the uncertainty of an agent in a social situation. Then we can define an 'epistemic update' operation that represents the effect of a communicatory action, such as a public announcement, on the original model. For example, publicly announcing a true formula φ, converts the current model to a submodel in which φ is true at each state. Starting with [$\text{Pl}_0$07] and more recently [$\text{Ba}_4\text{Mo}_3$04, $\text{Ko}_4$03, vD00, $\text{Ge}_2$99a, vB06], logical systems have been developed with the intent to capture the dynamics of information in a social situation. Chapter 4 of Kooi's dissertation [$\text{Ko}_4$03] and the recent book [vDvHK$\text{o}_4$07] contain a thorough discussion of the current state of affairs.

These logics use **PDL** style operators to represent an epistemic update. For example, if $!\varphi$ is intended to mean a public announcement of φ, then $\langle !\varphi\rangle \mathbf{K}_i\psi$ is intended to mean that after φ is publicly announced, agent i knows ψ. From this point of view, our communication modality $\Diamond$ can be understood as existentially quantifying over a sequence of *private* epistemic updates. However, there are some important differences between the semantics presented in this paper and the semantics found in the dynamic epistemic logic literature. First of all, in our semantics, communication is limited by the communication graph. Secondly, we do not consider general epistemic updates as is common in the literature, but rather study a specific type of epistemic update and its connection with a communication graph. Most important is the fact that the history of communications plays a key role in the definition of knowledge in this paper. The general approach of dynamic epistemic semantics is to define update operations mapping Kripke structures to other Kripke structures intended to represent the effect of an epistemic update on the first Kripke structure. For example, a public announcement of φ selects the submodel of a Kripke structure in which φ is true at every state. The definition of knowledge after an epistemic update is the usual definition, i.e., φ is known by i at state w if φ is true in all states that i considers possible from state w in the updated Kripke structure.

Floris Roelofsen introduces communication graphs to the dynamic epistemic logic setting in [Ro$_0$05]. The framework is more general than the one presented in this paper in three respects. First, the communication graph is a relation on the collection of subsets of $\mathcal{A}$, where an edge between $\mathcal{B}_1 \subseteq \mathcal{A}$ and $\mathcal{B}_2 \subseteq \mathcal{A}$ means that group $\mathcal{B}_1$ can communicate with group $\mathcal{B}_2$. Thus a communication event in this framework is a tuple $(\mathcal{B}_1, \mathcal{B}_2, \varphi)$ intended to mean that group $\mathcal{B}_1$ sends a message whose content is φ to group $\mathcal{B}_2$; and a precondition for an event $(\mathcal{B}_1, \mathcal{B}_2, \varphi)$ to take place is that φ is common knowledge among group $\mathcal{B}_1$.[9] Second, there is no assumption that messages between groups be restricted to ground formulas. Finally, Roelofsen does not assume that the communication graph is common knowledge. Therefore, there may be messages that can "update" the agent's view of the communication graph. So, should our models be viewed as an interesting special case of these more general model? The answer to this question is not straightforward as the history of communication plays a crucial role in the semantics presented in this paper. A full comparison between these two approaches can be found in [Ho$_7$06] (cf. [vBPa$_0$06, vBGe$_2$Pa$_0$07] for a comparison between history based semantics and dynamic epistemic semantics).

4 Conclusions

In this paper we have introduced a logic of knowledge and communication. Communication among agents is restricted by a communication graph, and idealized in the sense that the agents are unaware when their knowledge base is being accessed. We have shown that the communication graph is characterized by the validities of formulas in models based on that communication graph, and that our logic is decidable.

We are now moving on to discuss future work. Standard questions such as finding an elegant complete axiomatization will be studied. The semantics described in Section 2 rests on some strong underlying assumptions. Below we briefly sketch how to remove some of these assumptions.

One-way communication

As discussed in the introduction, an edge from i to j means that i can read j's website *without* j knowing that its website is being read. Thus a communication event (i, j, φ) only changes i's knowledge. This can be formally verified by noting that if H and $H; (i, j, \varphi)$ are w-legal histories, then by the definition of the λ_j function, $\lambda_j(H) = \lambda_j(H; (i, j, \varphi))$. Thus

[9] We *could* of course consider other cases where some members of $\mathcal{B}_1$ communicate with some members of $\mathcal{B}_2$.

$(w, H) \sim_j (v, H')$ iff $(w, H; (i, j, \varphi)) \sim_j (v, H')$ and so j's knowledge is unchanged by the presence of the event (i, j, φ). We can model *conscious communication* by changing the definition of the local view function. Define the λ_i^* as follows: given a history H, let $\lambda_i^*(H)$ map events of the form (i, j, φ) and (j, i, φ) to themselves and all other events in which i does not occur in the first two components to the null event.

Starting with theories

Another natural extension is to consider situations in which agents have a preference over which information they will read from another agent's website. Thus for example, if one hears that an English Ph.D. student and his adviser recently had a meeting, then one is justified in assuming that they probably did not discuss the existence of non-recursive sets, even though the adviser may conceivably know this fact. Given that this preference over the formulas under discussion among different groups of agents is common knowledge, each agent can regard some (legal) histories as being more or less likely than other (legal) histories. From this ordering over histories, we can define a *defeasible* knowledge operator for each agent. The operator is defeasible in the sense that agents may be wrong, i.e., it *is* after all possible that the English student and his adviser actually spent the meeting discussing the fact that there must be a non-recursive set.

References

[AivBPH07] M. Aiello, J.F.A.K. van Benthem & I.E. Pratt-Hartmann, eds. *Handbook of Spatial Logics*. Springer, 2007.

[Au76] R.J. Aumann. Agreeing to disagree. *Annals of Statistics* 4(6):1236–1239, 1976.

[Ba$_1$85] M. Bacharach. Some extensions of a claim of Aumann in an axiomatic model of knowledge. *Journal of Economic Theory* 37(1):167–190, 1985.

[Ba$_4$Mo$_3$04] A. Baltag & L. Moss. Logics for epistemic programs. *Synthese* 139(2):165–224, 2004.

[Bo$_0$Ne97] G. Bonanno & K. Nehring. Agreeing to disagree: A survey, 1997. Preprint.

[Ca$_5$83] J.A.K. Cave. Learning to agree. *Economics Letters* 12(2):147–152, 1983.

[$\mathrm{Ch}_1\mathrm{Ko}_0\mathrm{Po}_0$06] Z. Chatzidakis, P. Koepke & W. Pohlers, eds. *Logic Colloquium '02*, *Lecture Notes in Logic* 27. Association for Symbolic Logic, 2006.

[$\mathrm{Ch}_4$99] M.S.-Y. Chwe. Structure and strategy in collective action. *American Journal of Sociology* 105(1):128–156, 1999.

[$\mathrm{Ch}_4$00] M.S.-Y. Chwe. Communication and coordination in social networks. *Review of Economic Studies* 67(1):1–16, 2000.

[Fa+95] R. Fagin, J. Halpern, Y. Moses & M. Vardi. *Reasoning about Knowledge.* The MIT Press, 1995.

[$\mathrm{Fi}_2$91a] M. Fitting. Many-valued modal logics. *Fundamenta Informaticae* 15(3–4):235–254, 1991.

[$\mathrm{Fi}_2$91b] M. Fitting. Many-valued modal logics II. *Fundamenta Informaticae* 17(1–2):55–73, 1992.

[$\mathrm{Ge}_0\mathrm{Po}_1$82] J. Geanakoplos & H. Polemarchakis. We can't disagree forever. *Journal of Economic Theory* 28(1):192–200, 1982.

[$\mathrm{Ge}_2$99a] J. Gerbrandy. *Bisimulations on planet Kripke.* Ph.D. thesis, Universiteit van Amsterdam, 1999. *ILLC Publications* DS-1999-01.

[$\mathrm{Go}_4\mathrm{Ho}_2\mathrm{Ve}_1$06] G. Governatori, I. Hodkinson & Y. Venema, eds. *Advances in Modal Logic 6.* College Publications, 2006.

[$\mathrm{He}_0$96] A. Heifetz. Comment on consensus without common knowledge. *Journal of Economic Theory* 70(1):273–277, 1996.

[$\mathrm{Ho}_7$06] T. Hoshi. The logic of communication graphs for group communication and the dynamic epistemic logic with a future operator, 2006. Unpublished manuscript.

[$\mathrm{Ko}_4$03] B. Kooi. *Knowledge, Chance and Change.* Ph.D. thesis, University of Groningen, 2003. *ILLC Publications* DS-2003-01.

[$\mathrm{Kr}_0$96] P. Krasucki. Protocols forcing consensus. *Journal of Economic Theory* 70(1):266–272, 1996.

[Le_0+05] J.A. Leite, A. Omicini, P. Torroni & P. Yolum, eds. *Declarative Agent Languages and Technologies II, Second International Workshop, DALT 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers*, *Lecture Notes in Computer Science* 3476. Springer, 2005.

[$\mathrm{Le_2 69}$] D. Lewis. *Convention.* Harvard University Press, 1969.

[$\mathrm{Mo_2 92}$] Y. Moses, ed. *TARK '92: Proceedings of the 4th conference on Theoretical aspects of reasoning about knowledge.* Morgan Kaufmann, 1992.

[$\mathrm{Mo_3 Pa_5 92}$] L. Moss & R. Parikh. Topological reasoning and the logic of knowledge. In [$\mathrm{Mo_2 92}$, pp. 95–105].

[$\mathrm{Pa_0 Pa_5 05}$] E. Pacuit & R. Parikh. The logic of communication graphs. In [$\mathrm{Le_0}$+05, pp. 256–269].

[$\mathrm{Pa_5 95}$] R. Parikh, ed. *Logics of Programs, Conference, Brooklyn College, June 17–19, 1985, Proceedings*, *Lecture Notes in Computer Science* 193. Springer, 1985.

[$\mathrm{Pa_5 Kr_0 90}$] R. Parikh & P. Krasucki. Communication, consensus, and knowledge. *Journal of Economic Theory* 52(1):178–189, 1990.

[$\mathrm{Pa_5 Mo_3 St_4 07}$] R. Parikh, L. Moss & C. Steinsvold. Topology and epistemic logic. In [AivBPH07, pp. 299–341].

[$\mathrm{Pa_5 Ra_2 85}$] R. Parikh & R. Ramanujam. Distributed processes and the logic of knowledge. In [$\mathrm{Pa_5 95}$, pp. 256–268].

[$\mathrm{Pa_5 Ra_2 03}$] R. Parikh & R. Ramanujam. A knowledge based semantics of messages. *Journal of Logic, Language and Information* 12(4):453–467, 2003.

[$\mathrm{Pl_0 07}$] J. Plaza. Logics of public communications. *Synthese* 158:165–179, 2007.

[$\mathrm{Ra_4 89}$] Z.W. Ras, ed. *Methodologies for Intelligent Systems.* North-Holland, 1989.

[$\mathrm{Ra_5 Ma_5 93}$] H. Rasiowa & V.W. Marek. Mechanical proof systems for logic II, consensus programs and their processing. *Journal of Intelligent Information Systems* 2(2):149–164, 1993.

[$\mathrm{Ra_5 Ma_5 89}$] H. Rasiowa & W. Marek. On reaching consensus by groups of intelligent agents. In [$\mathrm{Ra_4 89}$, pp. 234–243].

[$\mathrm{Ro_0 05}$] F. Roelofsen. *Exploring logical perspectives on distributed information and its dynamics.* Master's thesis, Universiteit van Amsterdam, 2005. *ILLC Publications* MoL-2005-05.

[$Sa_2 07$] D. Samet, ed. *Theoretical Aspects of Rationality and Knowledge. Proceedings of the XIth Conference.* UCL Presses, 2007. Forthcoming.

[vB06] J. van Benthem. 'One is a Lonely Number': On the logic of communication. In [$Ch_1 Ko_0 Po_0 06$, pp. 96–129].

[$vBGe_2 Pa_0 07$] J. van Benthem, J. Gerbrandy & E. Pacuit. Merging frameworks for interaction: ETL and DEL. In [$Sa_2 07$]. Forthcoming.

[$vBPa_0 06$] J. van Benthem & E. Pacuit. The tree of knowledge in action: Towards a common perspective. In [$Go_4 Ho_2 Ve_1 06$, pp. 87–106].

[vD00] H. van Ditmarsch. *Knowledge Games.* Ph.D. thesis, University of Groningen, 2000. *ILLC Publications* DS-2000-06.

[$vDvHKo_4 07$] H. van Ditmarsch, W. van der Hoek & B. Kooi. *Dynamic Epistemic Logic*, *Synthese Library* 337. Springer, 2007.

Epistemic Foundations for Backward Induction: An Overview

Andrés Perea

Department of Quantitative Economics
Universiteit Maastricht
6200 MD Maastricht, The Netherlands
a.perea@ke.unimaas.nl

Abstract

In this survey we analyze and compare various sufficient epistemic conditions for backward induction that have been proposed in the literature. To this purpose we present a simple epistemic base model for games with perfect information, and express the conditions of the different models in terms of our base model. This will enable us to explictly analyze the differences and similarities between the various sufficient conditions for backward induction.

1 Introduction

Backward induction constitutes one of the oldest concepts in game theory. Its algorithmic definition, which goes back at least to [Ze13], seems so natural at first sight that one might be tempted to argue that every player "should" reason in accordance with backward induction in every game with perfect information. However, on a decision theoretic level the concept is no longer as uncontroversial as it may seem. The problem is that the backward induction algorithm, when applied from a certain decision node on, completely ignores the history that has led to this decision node, as it works from the terminal nodes towards this decision node. At the same time, the beliefs that the player at this decision node has about his opponents' future behavior may well be affected by the history he has observed so far. For instance, a player who observes that an opponent has not chosen in accordance with backward induction in the past may have a valid reason to believe that this same opponent will continue this pattern in the game that lies ahead. However, such belief revision policies are likely to induce choices that contradict backward induction. We therefore need to impose some non-trivial conditions on the players' belief revision policies in order to arrive at backward induction.

During the last decade or so, the game-theoretic literature has provided us with various epistemic models for dynamic games in which sufficient epistemic conditions for backward induction have been formulated. The objective of this survey is to discuss these conditions individually, and to explicitly compare the different conditions with each other. The latter task is particularly difficult since the literature exhibits a large variety of epistemic models, each with its own language, assumptions and epistemic operators. Some models are syntactic while others are semantic, and among the semantic models some are based on the notion of states of the world while others use types instead. As to the epistemic operators, some models apply knowledge operators while others use belief operators, and there is also a difference with respect to the "timing" of these operators. Are players entitled to revise their knowledge or belief during the course of the game, and if so, at which instances can they do so? Different models provide different answers to these, and other, questions.

As to overcome these problems we present in Section 2 an epistemic base model, which will be used as a reference model throughout this overview. In Section 3 we then provide for each of the papers to be discussed a brief description of the model, followed by an attempt to formulate its epistemic conditions for backward induction in terms of our base model. By doing so we formulate all sufficient conditions for backward induction in the same base model, which makes it possible to explicitly analyze the differences and similarities between the various conditions.

Finally, a word about the limitations of this paper. In this survey, we restrict attention to epistemic conditions that lead to the *backward induction strategies for all players.* There are alternative models that lead to the *backward induction outcome,* but not necessarily to the backward induction strategy for each player. For instance, [$\text{Ba}_7\text{Si}_1$02] and [$\text{Br}_1\text{Fr}_2\text{Ke}_1$04] provide epistemic models for extensive form rationalizability [$\text{Pe}_0$84, $\text{Ba}_7$97] and iterated maximal elimination of weakly dominated strategies, respectively, which always lead to the backward induction outcome in every generic game with perfect information, but not necessarily to the backward induction strategy profile. We also focus exclusively on sufficient conditions that apply to *all* generic games with perfect information. There are other interesting papers that deal with the logic of backward induction in *specific* classes of games, such as Rosenthals's centipede game [$\text{Ro}_3$81] and the finitely repeated prisoner's dilemma. See, among others, [$\text{Bi}_1$87, $\text{St}_1$96, Au98, $\text{Ra}_0$98, $\text{Br}_3\text{Ra}_0$99, $\text{Ca}_2$00, Pr00]. We shall, however, not discuss these papers here. Even with the limitations outlined above, we do not claim to offer an exhaustive list of epistemic models for backward induction. We do believe, however, that the list of models treated here will give the reader a good impression of the various epistemic conditions for backward induction that exist in the literature.

2 An epistemic base model

2.1 Games with perfect information

A dynamic game is said to be with *perfect information* if every player, at each instance of the game, observes the opponents' moves that have been made until then. Formally, an *extensive form structure $\mathcal{S}$ with perfect information* consists of the following ingredients:

- First, there is a *rooted, directed tree* $T = (X, E)$, where X is a finite set of nodes, and $E \subseteq X \times X$ is a finite set of directed edges. The nodes represent the different situations that may occur during the game, and the edges (x, y) represent moves by players that carry the game from situation x to situation y. The root $x_0 \in X$ marks the beginning of the game. For every two nodes $x, y \in X$, there is at most one path $((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n))$ in E from x to y with $x_1 = x$, $y_n = y$, and $y_k = x_{k+1}$ for every $k \in \{1, \ldots, n-1\}$. We say that x precedes y (or, y follows x) if there is a path from x to y. Since x_0 is the root, there is for every $x \in X \backslash \{x_0\}$ a path from x_0 to x. A node $x \in X$ is called a *terminal node* if it is not followed by any other node in X. The set of terminal nodes is denoted Z, and represents the set of possible outcomes for the game.

- There is a finite set I of *players*, and a *move function* $m : X \backslash Z \to I$ which specifies for every non-terminal node x the player $m(x) \in I$ who has to move at x. For every player i, the set of nodes

$$H_i := \{x \in X \backslash Z \mid m(x) = i\}$$

  is called the set of *information sets*[1] for player i. Since every information set $h_i \in H_i$ corresponds to a single node it is assumed that a player, whenever it is his turn to move, knows exactly which node in the game tree has been reached. That is, the game has *perfect information.* The root x_0 is identified with the information set h_0, and by $H_i^* := H_i \cup \{h_0\}$ we denote the set of player i information sets, together with the beginning of the game. By $H = \bigcup_{i \in I} H_i$ we denote the set of all information sets.

- For every player i and information set $h_i \in H_i$, the set of edges

$$A(h_i) := \{(h_i, y) \mid y \in X,\ (h_i, y) \in E\}$$

  is called the set of *actions*, or moves, available at h_i. By $A = \bigcup_{h \in H} A(h)$ we denote the set of all actions.

[1] Note that in our restricted setting an information set consists of a single node. We could therefore also have used the term "non-terminal node" instead of "information set". In particular, the collection of all information sets coincides with the set of all non-terminal nodes.

We now turn to the definition of a strategy. Intuitively, a strategy for player i is a plan that describes what player i would do in every possible situation in the game where it is his turn to move. The formal definition of a strategy we shall employ coincides with the concept of a *plan of action*, as discussed in [Ru$_1$91]. The difference with the usual definition is that we require a strategy only to prescribe an action at those information sets that the same strategy does not avoid. Formally, let $\tilde{H}_i \subseteq H_i$ be a collection of player i information sets, not necessarily containing all information sets, and let $s_i : \tilde{H}_i \to A$ be a mapping prescribing at every $h_i \in \tilde{H}_i$ some available action $s_i(h_i) \in A(h_i)$. For a given information set $h \in H$, not necessarily belonging to player i, we say that s_i *avoids* h if on the path

$$((x_1, y_1), \ldots, (x_n, y_n))$$

from h_0 to h there is some node x_k in $\tilde{H}_i$ with $s_i(x_k) \neq (x_k, y_k)$. That is, the prescribed action $s_i(x_k)$ deviates from this path. Such a mapping $s_i : \tilde{H}_i \to A$ is called a *strategy* for player i if $\tilde{H}_i$ is exactly the collection of player i information sets not avoided by s_i. Obviously, every strategy s_i can be obtained by first prescribing an action at all player i information sets, that is, constructing a strategy in the classical sense, and then deleting from its domain those player i information sets that are avoided by it. For a given strategy $s_i \in S_i$, we denote by $H_i(s_i)$ the collection of player i information sets that are not avoided by s_i. Let S_i be the set of player i strategies. For a given information set $h \in H$ and player i, we denote by $S_i(h)$ the set of player i strategies that do not avoid h. Then, it is clear that a profile $(s_i)_{i \in I}$ of strategies reaches an information set h if and only if $s_i \in S_i(h)$ for all players i.

2.2 Preferences, beliefs and types

The basic assumption in our base model is that every player has a *strict*[2] preference relation over the terminal nodes, and holds at each of his information sets a conditional belief about the opponents' strategy choices and preference relations. In particular, we allow for the fact that players may revise their beliefs about the opponents' preferences as the game proceeds. In order to keep our model as "weak" as possible, we assume that this conditional belief can be expressed by a *set* of opponents' strategies and preference relations. This set represents the strategies and preference relations that the player deems *possible* at his information set. We thus do not consider probabilities, and it is therefore sufficient to specify the players' *ordinal* preferences over terminal nodes. Not only does a player hold first-order conditional beliefs about the opponents' choices and preferences,

[2] In the literature, it is not always assumed that players hold strict preferences over terminal nodes. We do so here for the sake of simplicity.

he also holds second-order conditional beliefs about the opponents' possible first-order beliefs at each of his information sets. A second-order belief may thus contain expressions of the form "player i considers it possible at information set h_i that player j considers it possible at information set h_j that player k chooses strategy s_k and has preference relation P_k". Recursively, one may define higher-order conditional beliefs for the players. A possible way to represent such hierarchies of conditional beliefs is by means of the following model.

Definition 2.1 (Epistemic base model)**.** Let $\mathcal{S}$ be an extensive form structure with perfect information. An *epistemic base model* for $\mathcal{S}$ is a tuple

$$\mathcal{M} = (T_i, P_i, B_i)_{i \in I}$$

where

(1) T_i is a set of *types* for player i;

(2) P_i is a function that assigns to every $t_i \in T_i$ some complete, strict and transitive preference relation $P_i(t_i)$ over the terminal nodes;

(3) B_i is a function that assigns to every $t_i \in T_i$ and every information set $h_i \in H_i^*$ some subset $B_i(t_i, h_i) \subseteq \prod_{j \neq i}(S_j(h_i) \times T_j)$.

Here, $B_i(t_i, h_i)$ denotes the set of opponents' strategy-type pairs which t_i deems possible at h_i. We denote by $B_i(t_i, h_i | S_{-i})$ the projection of $B_i(t_i, h_i)$ on $\prod_{j \neq i} S_j(h_i)$. That is, $B_i(t_i, h_i | S_{-i})$ is t_i's belief at h_i about the opponents' strategy choices. For any player $j \neq i$, we denote by $B_{ij}(t_i, h_i)$ the projection of $B_i(t_i, h_i)$ on the set $S_j(h_i) \times T_j$. Hence, $B_{ij}(t_i, h_i)$ is t_i's belief at h_i about player j's strategy-type pair.

From an epistemic base model, conditional beliefs of any order can be *derived.* For instance, type t_i's belief at h_i about player j's choice is given by the projection of $B_{ij}(t_i, h_i)$ on S_j. Let $B_{ij}(t_i, h_i | S_j)$ denote this projection, and let $B_{ij}(t_i, h_i | T_j)$ denote its projection on T_j. Then, type t_i's belief at h_i about player j's belief at h_j about player ℓ's choice is given by

$$\bigcup_{t_j \in B_{ij}(t_i, h_i | T_j)} B_{j\ell}(t_j, h_j | S_\ell).$$

In a similar fashion, higher-order beliefs can be derived.

2.3 Common belief

Let $\mathcal{M} = (T_i, P_i, B_i)_{i \in I}$ be an epistemic base model, and $E \subseteq \bigcup_{j \in I} T_j$ a set of types, or *event.* We say that type t_i *believes in* E at information set $h_i \in H_i^*$ if $B_{ij}(t_i, h_i | T_j) \subseteq E$ for all $j \neq i$. We say that t_i *initially believes* *in* E if t_i believes in E at h_0. Common belief in the event E is defined by the following recursive procedure:

$$\mathrm{B}_i^1(E) = \{t_i \in T_i \mid B_{ij}(t_i, h_i|T_j) \subseteq E \text{ for all } j \neq i \text{ and all } h_i \in H_i^*\}$$

for all $i \in I$, and

$$\mathrm{B}_i^{k+1}(E) = \{t_i \in T_i \mid B_{ij}(t_i, h_i|T_j) \subseteq B_j^k(E) \text{ for all } j \neq i \text{ and all } h_i \in H_i^*\}$$

for all $i \in I$ and all $k \geq 1$.

Definition 2.2 (Common belief)**.** A type $t_i \in T_i$ is said to respect *common belief* in the event E if $t_i \in E$ and $t_i \in \mathrm{B}_i^k(E)$ for all k.

Hence, t_i respects common belief in E if t_i belongs to E, believes throughout the game that opponents' types belong to E, believes throughout the game that opponents believe throughout the game that the other players' types belong to E, and so on. In particular, if we say that t_i respects common belief in E, this implies that t_i itself should belong to E. So, for instance, if we say that t_i respects common belief in the event that types never change their belief about the opponents' preference relations during the game, this implies that t_i itself never changes its belief about the opponents' preference relations. We realize that this is perhaps linguistically not correct, but we do so for the sake of brevity. Otherwise, we should have to write, throughout this paper, that t_i belongs to E, and respects common belief in E.

In most other epistemic models in the literature, the term "common belief" or "common knowledge" refers to the epistemic state of a *group* of players, rather than to the epistemic state of a single player, as we use it. That is, in most other models the expression "there is common belief in E" means that "all players believe that E holds, all players believe that all players believe that E holds, and so on." The reason for me to use an "individualistic" version of common belief is that we want to impose conditions on the beliefs of *one player only*, and see when such individual conditions lead to backward induction reasoning by that player. So, we take a single-player perspective in this paper, and choose the version of common belief accordingly. We realize this is an unusual approach, but it is well-suited for the purposes we have in mind.

Common initial belief in the event E is defined as follows:

$$\mathrm{IB}_i^1(E) = \{t_i \in T_i \mid B_{ij}(t_i, h_0|T_j) \subseteq E \text{ for all } j \neq i\}$$

for all $i \in I$, and

$$\mathrm{IB}_i^{k+1}(E) = \{t_i \in T_i \mid B_{ij}(t_i, h_0|T_j) \subseteq \mathrm{IB}_j^k(E) \text{ for all } j \neq i\}$$

for all $i \in I$ and all $k \geq 1$.

Definition 2.3 (Common initial belief)**.** A type $t_i \in T_i$ is said to respect *common initial belief* in the event E if $t_i \in E$ and $t_i \in \mathrm{IB}_i^k(E)$ for all k.

2.4 Belief in the opponents' rationality

All the epistemic foundations for backward induction to be discussed here make assumptions about the beliefs that players have about the rationality of their opponents. More precisely, all foundations require that players *initially* believe that each opponent chooses rationally at every information set. However, the various foundations differ as to how players would *revise* their beliefs upon observing that their initial belief about the opponents was incorrect. In order to express these different belief revision procedures in terms of our base model, we need the following definitions.

We first define what it means that a strategy is rational for a type at a given information set. For an information set $h_i \in H_i$, a strategy $s_i \in S_i(h_i)$, and an opponents' strategy profile $s_{-i} \in \prod_{j \neq i} S_j(h_i)$, let $z(s_i, s_{-i}|h_i)$ be the terminal node that would be reached from h_i if (s_i, s_{-i}) were to be executed by the players.

Definition 2.4 (Rationality at an information set)**.** A strategy s_i is *rational* for type t_i at information set $h_i \in H_i(s_i)$ if there is no $s_i' \in S_i(h_i)$ such that $P_i(t_i)$ ranks $z(s_i', s_{-i}|h_i)$ strictly over $z(s_i, s_{-i}|h_i)$ for all $s_{-i} \in B_i(t_i, h_i|S_{-i})$.

Hence, s_i is rational for t_i at h_i is there is no other strategy $s_i' \in S_i(h_i)$ that strictly dominates s_i, given the set of opponents' strategies that t_i deems possible at h_i.

We shall now define various restrictions on the beliefs that players have about the opponents' rationality. We need one more definition to this purpose. For a given type $t_i \in T_i$, information set $h_i \in H_i^*$, and some opponent's information set $h \in H \backslash H_i$ following h_i, we say t_i *believes h to be reached from h_i* if $B_i(t_i, h_i|S_{-i}) \subseteq S_{-i}(h)$. Here, $S_{-i}(h)$ is a short way to write $\prod_{j \neq i} S_j(h)$.

Definition 2.5 (Belief in the opponents' rationality)**.**

(1) Type t_i believes at information set $h_i \in H_i^*$ that player j chooses rationally at information set $h_j \in H_j$ if for every $(s_j, t_j) \in B_{ij}(t_i, h_i)$ it is true that s_j is rational for t_j at h_j.

(2) Type t_i *initially* believes in rationality at *all* information sets if t_i believes at h_0 that every opponent j chooses rationally at all $h_j \in H_j$.

(3) Type t_i *always* believes in rationality at *all future* information sets if t_i believes at every $h_i \in H_i^*$ that every opponent j chooses rationally at every $h_j \in H_j$ that follows h_i.

(4) Type t_i *always* believes in rationality at *future information sets that are believed to be reached* if t_i believes at every $h_i \in H_i^*$ that every opponent j chooses rationally at all those $h_j \in H_j$ following h_i which t_i believes to be reached from h_i.

(5) Type t_i *always* believes in rationality at *all future and parallel* information sets if t_i believes at every $h_i \in H_i^*$ that every opponent j chooses rationally at every $h_j \in H_j$ that does not precede h_i.

(6) Type t_i *always* believes in rationality at *all* information sets if t_i believes at every $h_i \in H_i^*$ that every opponent j chooses rationally at every $h_j \in H_j$.

Condition (6) is the strongest possible condition that can be imposed, since it requires a player, under all circumstances, to maintain his belief that his opponents have chosen rationally in the past, will choose rationally at any stage in the future, and would have chosen rationally at all foregone (i.e., parallel) information sets. In particular, a player is assumed to interpret every observed past move as being part of an opponent's strategy which is rational at all information sets. In other words, every past move is interpreted as a rational move.

Condition (5) is a weakening of (6). In condition (5), a player need not interpret every observed past move as a rational move, since he is no longer required to believe in the opponents' rationality at past information sets. That is, if a player observes an opponent's move which surprises him, then he may believe that this move was due to a mistake by the opponent. However, condition (5) still requires the player to believe that this same opponent will choose rationally all stages in the future, and would have chosen rationally at all foregone situations. Hence, in condition (5) an observed surprising move by an opponent should not be a reason for dropping the belief in this opponent's rationality at future and foregone situations.

Condition (3) is a weakening of (5), since it no longer requires that a player, after observing a surprising move by an opponent, believes that this opponent would have chosen rationally at foregone situations. However, the player is still assumed to believe that the opponent will, and would, choose rationally at all future situations, no matter whether he deems these future situations possible or not.

Condition (4), in turn, is a weakening of (3). In condition (4) a player, after observing a surprising move by an opponent, need not believe that this opponent would choose rationally at future situations which he does not deem possible. He is only required to believe that the opponent will choose rationally at future situations which he indeed believes could take place.

Condition (2) is a weakening of (3) but not of (4). In condition (2), a player believes, before anything has happened, that every opponent will, and would, choose rationally at all future situations, no matter whether he deems these situations possible or not. However, once the game is under way, the player is allowed to completely drop his belief in the opponents' rationality. Note than in condition (4), a player may initially believe that an opponent would not choose rationally at a certain information set if he initially believes that this information set will not be reached. Therefore, condition (2) is not a weakening of (4). Also, condition (4) is not a weakening of (2), so there is no logical implication betweens conditions (2) and (4).

In light of the definitions we have seen so far, we may thus construct phrases as "type t_i respects common belief in the event that all types initially believe in rationality at all information sets". Some of the epistemic foundations for backward induction, however, use a condition that cannot be expressed in this form, since it relies on a notion that is different from common belief. In order to formalize this condition, we consider the following recursive procedure:

$$\begin{aligned}\mathrm{FBSR}_i^1(h_i) = \{t_i \in T_i \mid\ & t_i \text{ believes at } h_i \text{ that every } j \neq i \text{ chooses} \\ & \text{rationally at all } h_j \text{ that follow } h_i\}\end{aligned}$$

for all $i \in I$ and all $h_i \in H_i^*$, and

$$\begin{aligned}\mathrm{FBSR}_i^{k+1}(h_i) = \{t_i \in T_i \mid\ & B_{ij}(t_i, h_i|T_j) \subseteq \mathrm{FBSR}_j^k(h_j) \text{ for all } j \neq i \\ & \text{and all } h_j \text{ that follow } h_i\}\end{aligned}$$

for all $i \in I$, $h_i \in H_i^*$ and $k \geq 1$.

Definition 2.6 (Forward belief in substantive rationality)**.** A type t_i is said to respect *forward belief in substantive rationality* if $t_i \in \mathrm{FBSR}_i^k(h_i)$ for all k and all $h_i \in H_i^*$.

That is, t_i respects forward belief in substantive rationality if t_i (1) always believes that every opponent is rational at every future information set, (2) always believes that every opponent, at every future information set, believes that every opponent is rational at every future information set, (3) always believes that every opponent, at every future information set, believes that every opponent, at every future information set, believes that every opponent is rational at every future information set, and so on.

The first condition above, namely that t_i always believes that every opponent is rational at every future information set, corresponds exactly to condition (3) of Definition 2.5. However, forward belief in substantive rationality is logically weaker than common belief in condition (3) of Definition 2.5. Consider, namely, a player i information set h_i and a player j information set h_j that precedes h_i. Then, common belief in condition (3) requires that player i believes at h_i that player j believes at h_j that player i chooses rationally at h_i, since h_i follows h_j. On the other hand, forward belief in substantive rationality does not require this, as it only restricts the belief that player i has at h_i about the beliefs that opponents have at information sets *following* h_i, but not preceding h_i. At the same time, it can be verified that forward belief in substantive rationality implies common *initial* belief in condition (3).

We also present a weaker version of forward belief in rationality, which we call forward belief in *material* rationality. Let $H_j(t_i, h_i)$ be the set of those player j information sets h_j following h_i which t_i believes to be reached from h_i. Consider the following recursive procedure:

$$\mathrm{FBMR}_i^1(h_i) = \{t_i \in T_i \mid t_i \text{ believes at } h_i \text{ that every } j \neq i \text{ chooses rationally at all } h_j \text{ in } H_j(t_i, h_i)\}$$

for all $i \in I$ and all $h_i \in H_i^*$, and

$$\mathrm{FBMR}_i^{k+1}(h_i) = \{t_i \in T_i \mid B_{ij}(t_i, h_i|T_j) \subseteq \mathrm{FBMR}_j^k(h_j) \text{ for all } j \neq i \text{ and all } h_j \text{ in } H_j(t_i, h_i)\}$$

for all $i \in I$, $h_i \in H_i^*$ and $k \geq 1$.

Definition 2.7 (Forward belief in material rationality)**.** A type t_i is said to respect *forward belief in material rationality* if $t_i \in \mathrm{FBMR}_i^k(h_i)$ for all k and all $h_i \in H_i^*$.

The crucial difference with forward belief in substantive rationality is thus that a type is only required to believe his opponents to choose rationally at future information sets *which he believes to be reached.* And a type is only required to believe that the opponents' types believe so at future information sets which he believes to be reached, and so on.

The condition in the first step of the recursive procedure, namely that t_i believes at h_i that every opponent j chooses rationally at all h_j in $H_j(t_i, h_i)$, corresponds exactly to condition (4) of Definition 2.5. However, by the same argument as above for forward belief in substantive rationality, it can be verified that forward belief in material rationality is logically weaker than common belief in condition (4) of Definition 2.5. On the other hand, forward belief in material rationality implies common *initial* belief in condition (4).

3 Epistemic foundations for backward induction

In this section we provide an overview of various epistemic foundations that have been offered in the literature for backward induction. A comparison between these foundations is difficult, since the models used by these foundations differ on many aspects.

A first important difference lies in the way the players' beliefs about the opponents are expressed. Some models express the players' beliefs *directly* by means of logical propositions in some formal language. Other models represent the players' beliefs *indirectly* by a set of states of the world, and assign to each state and every player some strategy choice for this player, together with a belief that the player holds at this state about the state of the world. From this model we can derive the higher-order beliefs that players hold about the opponents' choices and beliefs. There are yet some other models that represent the players' beliefs indirectly by means of types, and assign to every type some belief about the other players' choices and types. Similarly to the previous approach, the players' higher-order beliefs can be derived from this model. We refer to these three approaches as the *syntactic model,* the *state-based semantic model* and the *type-based syntactic model.* Note that our base model from the previous section belongs to the last category. This choice is somewhat arbitrary, since we could as well have chosen a syntactic or state-based semantic base model.

Even within the state-based semantic model, the various papers differ on the precise formalization of the beliefs that players have about the state of the world. Similarly, within the type-based model different papers use different belief operators expressing the players' beliefs about the opponents' choices and types.

Finally, some models impose additional conditions on the extensive form structure, such as one information set per player, or the presence of only two players, whereas other papers do not.

In spite of these differences, all foundations have two aspects in common. First, all models provide a theorem, say Theorem A, which gives a sufficient condition for backward induction. Hence, Theorem A states that if player i's belief revision procedure about the other players' choices, preferences and beliefs satisfies some condition **BR**, then his unique optimal choice is his backward induction choice. Secondly, all models guarantee that this sufficient condition **BR** is possible. That is, each paper provides a second result, say Theorem B, which states that for every player i there is some model in which player i's belief revision procedure satisfies condition **BR**. As we shall see, the various foundations differ in the sufficient condition **BR** that is being employed.

In order to explicitly compare the different foundations for backward induction, we attempt to express the various conditions **BR** used by the different models in terms of our base model. By doing so, we express the Theorems A and B used by the various foundations in the following standardized form:

Theorem A. Let $\mathcal{S}$ be an extensive form structure with perfect information, and let $\mathcal{M} = (T_j, P_j, B_j)_{j \in I}$ be an epistemic base model for $\mathcal{S}$. Let

$(\tilde{P}_j)_{j\in I}$ be a profile of strict preference relations over the terminal nodes. If type $t_i \in T_i$ has preference relation $\tilde{P}_i$, and if t_i's conditional belief vector about the opponents' strategy choices and types satisfies condition **BR,** then there is a unique strategy that is rational for t_i at all information sets, namely his backward induction strategy in the game given by $(\tilde{P}_j)_{j\in I}$.

Theorem B. Let $\mathcal{S}$ be an extensive form structure with perfect information, and let i be a player. Then, there is some epistemic base model $\mathcal{M} = (T_j, P_j, B_j)_{j\in I}$ for $\mathcal{S}$ and some type $t_i \in T_i$ such that t_i's conditional belief vector satisfies **BR.**

In the overview that follows, we provide a brief description of every model, identify the condition **BR** that is being used, and explain how this condition may be expressed in terms of our base model. The models are put in alphabetical order.

3.1 Asheim's model

Asheim uses a type-based semantic model, restricted to the case of two players, in which the players' beliefs are modelled by lexicographic probability distributions [As02]. Formally, an Asheim model is given by a tuple

$$\mathcal{M} = (T_i, v_i, \lambda_i)_{i\in I}$$

where T_i is a finite set of types, v_i is a function that assigns to every t_i some von Neumann-Morgenstern utility function $v_i(t_i)$ over the set of terminal nodes, and λ_i is a function that assigns to every type t_i some lexicographic probability system $\lambda_i(t_i)$ on $S_j \times T_j$ with full support on S_j. Such a *lexicographic probability system* $\lambda_i(t_i)$ is given by a vector $(\lambda_i^1(t_i), \ldots, \lambda_i^{K_i(t_i)}(t_i))$ of probability distributions on $S_j \times T_j$. The interpretation is that $\lambda_i^1(t_i), \ldots, \lambda_i^{K_i(t_i)}(t_i)$ represent different degrees of beliefs, and that the kth degree belief $\lambda_i^k(t_i)$ is infinitely more important than the $(k+1)$st degree belief $\lambda_i^{k+1}(t_i)$, without completely discarding the latter. The lexicographic probability system $\lambda_i(t_i)$ induces in a natural way first-order conditional beliefs about player j's choices, as defined in our base model. Namely, for every $h_i \in H_i^*$, let $k_i(t_i, h_i)$ be the first k such that $\lambda_i^k(t_i)$ assigns positive probability to some strategy $s_j \in S_j(h_i)$, and let $\hat{B}_{ij}(t_i, h_i) \subseteq S_j(h_i)$ be the set of strategies in $S_j(h_i)$ to which $\lambda_i^{k_i(t_i,h_i)}(t_i)$ assigns positive probability. Then, t_i induces the conditional belief vector $(\hat{B}_{ij}(t_i, h_i))_{h_i\in H_i^*}$ about player j's strategy choice. For every h_i, let $\hat{T}_{ij}(t_i, h_i) \subseteq T_j$ be the set of types to which $\lambda_i^{k_i(t_i,h_i)}(t_i)$ assigns positive probability. Then, the induced second-order belief of t_i at h_i about player j's belief at h_j about player i's choice is given by the union of the sets $\hat{B}_{ji}(t_j, h_j)$ with $t_j \in \hat{T}_{ij}(t_i, h_i)$.

Similarly, higher-order beliefs about strategy choices can be derived from Asheim's model.

In Asheim's model, a strategy s_i is called rational for type $t_i \in T_i$ at information set h_i if s_i is optimal with respect to the utility function $v_i(t_i)$ and the lexicographic probability system $\lambda_i(t_i|h_i)$, where $\lambda_i(t_i|h_i)$ denotes the conditional of the lexicographic probability system $\lambda_i(t_i)$ on $S_j(h_i) \times T_j$. In particular, if s_i is rational for t_i at h_i then s_i is rational with respect to the preference relation $\hat{P}_i$ and the set-valued belief $\hat{B}_{ij}(t_i, h_i)$, as defined above, where $\hat{P}_i$ is the preference relation on terminal nodes induced by $v_i(t_i)$.

Asheim's sufficient condition for backward induction is based on the notion of *admissible subgame consistency.* A type t_i in an Asheim model is said to be *admissible subgame consistent* with respect to a given profile $(\tilde{v}_j)_{j \in I}$ of utility functions if (1) $v_i(t_i) = \tilde{v}_i$, and (2) for every $h_i \in H_i^*$, the probability distribution $\lambda_i^{k_i(t_i,h_i)}(t_i)$ only assigns positive probability to strategy-type pairs (s_j, t_j) such that s_j is rational for t_j at all $h_j \in H_j$ that follow h_i. In terms of our base model, this condition can be expressed as: (1') $P_i(t_i) = \tilde{P}_i$, and (2') t_i always believes in rationality at all future information sets. In fact, condition (2') is weaker than condition (2) since the notion of rationality in (2') is weaker than the notion of rationality in (2), but condition (2') would have sufficed to prove Asheim's theorem on backward induction.

In Proposition 7, Asheim shows that if a type t_i respects common certain belief in the event that types are admissible subgame consistent with respect to $(\tilde{v}_j)_{j \in I}$, then t_i has a unique strategy that is rational at all information sets, namely his backward induction strategy with respect to $(\tilde{v}_j)_{j \in I}$. Here, "certain belief in an event E" means that type t_i, in each of his probability distributions $\lambda_i^k(t_i)$, only assigns positive probability to types in E. In terms of our base model, this means that the type believes the event E at each of his information sets. In Proposition 8, Asheim shows that common certain belief in admissible subgame consistency is possible. Expressed in terms of our base model, Asheim's sufficient condition for backward induction may thus be written as follows:

Asheim's condition BR: Type t_i respects common belief in the events that (1) types hold preference relations as specified by $(\tilde{P}_j)_{j \in I}$, and (2) types always believe in rationality at all future information sets.

3.2 Asheim & Perea's model

In [AsPe$_2$05], Asheim and Perea propose a type-based semantic model that is very similar to the model from Section 3.1. Attention is restricted to two-player games, and an Asheim-Perea model corresponds to a tuple

$$\mathcal{M} = (T_i, v_i, \lambda_i, \ell_i)_{i \in I},$$

where T_i, v_i and λ_i are as in Asheim's model, and ℓ_i is a function that to every type t_i and event $E \subseteq S_j \times T_j$ assigns some number $\ell_i(t_i, E) \in \{1, \ldots, K_i(t_i)\}$. (Recall that $K_i(t_i)$ denotes the number of probability distributions in $\lambda_i(t_i)$). The interpretation of ℓ_i is that $\ell_i(t_i, E)$ specifies the number of probability distributions in $\lambda_i(t_i)$ that are to be used in order to derive the conditional lexicographic probability system of $\lambda_i(t_i)$ on E. For instance, if player i observes that his information set h_i has been reached, this would correspond to the event $E = S_j(h_i) \times T_j$. In this case, player i would use the first $\ell_i(t_i, E)$ probability distributions in $\lambda_i(t_i)$ in order to form his conditional belief about j upon observing that h_i has been reached. Two extreme cases are $\ell_i(t_i, E) = k_i(t_i, h_i)$, where player i would only use the first probability distribution in $\lambda_i(t_i)$ that assigns positive probability to some player j strategy in $S_j(h_i)$, and $\ell_i(t_i, E) = K_i(t_i)$, where player i would use the full lexicographic probability system $\lambda_i(t_i)$ to form his conditional belief upon observing that h_i is reached. Recall that $k_i(t_i, h_i)$ is the first k such that $\lambda_i^k(t_i)$ assigns positive probability to some strategy $s_j \in S_j(h_i)$.

The sufficient condition for backward induction is based on the event that *types induce for every opponent's type a sequentially rational behavior strategy.* Consider a type t_i, and let $T_j^{t_i}$ be the set of types to which the lexicographic probability system $\lambda_i(t_i)$ assigns positive probability (in some of its probability distributions). Asheim and Perea assume that for every $t_j \in T_j^{t_i}$ and every $s_j \in S_j$, the lexicographic probability system $\lambda_i(t_i)$ assigns positive probability to (s_j, t_j). For every information set $h_j \in H_j$ and action $a \in A(h_j)$, let $S_j(h_j, a)$ be the set of strategies in $S_j(h_j)$ that select action a at h_j. Define for every type $t_j \in T_j^{t_i}$, $h_j \in H_j$ and $a \in A(h_j)$

$$\sigma_j^{t_i|t_j}(h_j)(a) := \frac{\lambda_i^k(t_i)(S_j(h_j, a) \times \{t_j\})}{\lambda_i^k(t_i)(S_j(h_j) \times \{t_j\})},$$

where k is the first number such that $\lambda_i^k(t_i)(S_j(h_j) \times \{t_j\}) > 0$. The vector

$$\sigma_j^{t_i|t_j} = (\sigma_j^{t_i|t_j}(h_j)(a))_{h_j \in H_j, a \in A(h_j)}$$

is called the *behavior strategy induced by t_i for t_j*. My interpretation of $\sigma_j^{t_i|t_j}(h_j)(a)$ is that it describes t_i's conditional belief about t_j's action choices at future and parallel information sets. Let me explain. Consider an information set h_i for player i and an information set h_j for player j which either follows h_i or is parallel to h_i. Then, my interpretation of $\sigma_j^{t_i|t_j}(h_j)(a)$ is that type t_i believes at h_i that type t_j at information h_j chooses action a with probability $\sigma_j^{t_i|t_j}(h_j)(a)$. Namely, the information that the game has reached h_i does not give type t_i additional information about the action choice of t_j at h_j, and hence $\sigma_j^{t_i|t_j}(h_j)$ provides an intuitive candidate for the conditional belief of t_i at h_i about t_j's behavior at h_j.

However, if h_j precedes h_i, then $\sigma_j^{t_i|t_j}(h_j)(a)$ does not necessarily describe t_i's belief at h_i about t_j's action choice at h_j. In this case, there is namely a unique action a^* at h_j that leads to h_i, whereas $\sigma_j^{t_i|t_j}(h_j)(a^*)$ may be less than one (in fact, may be zero). On the other hand, it should be clear that t_i should believe (with probability 1) at h_i that t_j has chosen a^* at h_j, since it is the only action at h_j that leads to h_i. Hence, in this case $\sigma_j^{t_i|t_j}(h_j)(a)$ cannot describe t_i's belief at h_i about t_j's choice at h_j.

For every information set $h_j \in H_j$, let $\sigma_j^{t_i|t_j}|_{h_j}$ be the behavioral strategy that assigns probability one to all player j actions preceding h_j, and coincides with $\sigma_j^{t_i|t_j}$ otherwise. The induced behavior strategy $\sigma_j^{t_i|t_j}$ is said to be *sequentially rational for* t_j if at every information set $h_j \in H_j$, the behavior strategy $\sigma_j^{t_i|t_j}|_{h_j}$ only assigns positive probability to strategies in $S_j(h_j)$ that are rational for t_j at h_j (in the sense of Asheim's model above). Type t_i is said to induce for every opponent's type a sequentially rational behavior strategy if for every $t_j \in T_j^{t_i}$ it is true that $\sigma_j^{t_i|t_j}$ is sequentially rational for t_j. As we have seen above, $\sigma_j^{t_i|t_j}$ represents for every $h_i \in H_i^*$ type t_i's conditional belief at h_i about player j's behavior at future and parallel information sets. The requirement that $\sigma_j^{t_i|t_j}$ always be sequentially rational for t_j thus means that t_i always believes in rationality at all future and parallel information sets.

In Proposition 11, Asheim and Perea show that if a type t_i respects common certain belief in the events that (1) types have utility functions as specified by $(\tilde{v}_j)_{j\in I}$, and (2) types induce for every opponent's type a sequentially rational behavior strategy, then t_i has a unique strategy that is rational at all information sets, namely his backward induction strategy with respect to $(\tilde{v}_j)_{j\in I}$. The existence of such types follows from their Proposition 4 and the existence of a sequential equilibrium. In terms of our base model, Asheim and Perea's sufficient condition may thus be stated as follows:

Asheim & Perea's condition BR: Type t_i respects common belief in the events that (1) types hold preference relations as specified by $(\tilde{P}_j)_{i\in I}$, and (2) types always believe in rationality at all future and parallel information sets.

3.3 Aumann's model

Aumann proposes a state-based semantic model for extensive form structures with perfect information [Au95]. An Aumann model is a tuple

$$\mathcal{M} = (\Omega, (B_i, f_i, v_i)_{i\in I})$$

where Ω represents the set of states of the world, B_i is a function that assigns to every state $\omega \in \Omega$ some subset $B_i(\omega)$ of states, f_i is a function that assigns to every state ω some strategy $f_i(\omega) \in S_i$, and v_i is a function that assigns to every ω some von Neumann-Morgenstern utility function $v_i(\omega)$ on the set of terminal nodes. The functions B_i must have the property that $\omega \in B_i(\omega)$ for all ω, and for all $\omega, \omega' \in \Omega$ it must hold that $B_i(\omega)$ and $B_i(\omega')$ are either identical, or have an empty intersection. Hence, the set $\{B_i(\omega)|\omega \in \Omega\}$ is a partition of Ω. The interpretation is that at state ω, player i believes that the true state is in $B_i(\omega)$. (In fact, Aumann uses the term "knows" rather than "believes"). The functions f_i and v_i must be measurable with respect to B_i, meaning that $f_i(\omega') = f_i(\omega)$ whenever $\omega' \in B_i(\omega)$, and similarly for v_i. The reason is that player i cannot distinguish between states ω and ω', and hence his choice and preferences must be the same at both states.

It is problematic, however, to formally translate this model into conditional beliefs of our base model. Consider, for instance, a game with three players, in which players 1, 2 and 3 sequentially choose between Stay and Leave, and where Leave terminates the game. Consider a state ω where $f_1(\omega) = \mathsf{Leave}$ and $B_2(\omega) = \{\omega\}$. Then, at player 2's information set, player 2 must conclude that the state cannot be ω, but must be some state ω' with $f_1(\omega') = \mathsf{Stay}$. However, there may be many such states ω', and hence it is not clear how player 2 should revise his belief about the state at his information set. Since his revised belief about the state will determine his revised belief about player 3's choice, it is not clear how to explicitly define player 2's revised belief about player 3's choice from Aumann's model.

At the same time, Aumann's Theorem A provides sufficient conditions for backward induction, and hence Aumann's model must at least implicitly impose some restrictions on the players' belief revision procedures, since otherwise backward induction could not be established. The main task in this subsection will be to identify these implicit assumptions about the belief revision procedures, and incorporate these as explicit restrictions in our base model. Since such an identification is a rather subjective procedure, this approach will eventually lead to a model which is a subjective interpretation of Aumann's model.

The model as proposed by Aumann is essentially a static model, since for every state ω and every player i, his belief $B_i(\omega)$ is only defined at a single moment in time. My interpretation of these static beliefs is that players, upon observing that one of their information sets has been reached, do not revise more than "strictly necessary". In fact, the only beliefs that *must* be revised by player i when finding out that his information set h_i has been reached are, possibly, his beliefs about the opponents' choices at information sets preceding h_i. That is, if player 2 in the example above finds out that player 1 has chosen Stay, then this should not be a reason to change his belief about player 3's choice. Even stronger, we interpret Aumann's static beliefs in this game as beliefs in which player 2 *only* changes his belief about player 1's choice, while maintaining all his other beliefs, including his beliefs about the opponents' beliefs. Hence, if we interpret Aumann's model in this way, and express it accordingly in terms of our base model, then every type is supposed to never revise his belief about the opponents' choices and the opponents' beliefs at future and parallel information sets. A type t_i, when arriving at some information set h_i, may only revise his belief about the opponents' *choices* at information sets that precede h_i (but not about their types). For further reference, we call this condition the "no substantial belief revision condition".

The sufficient condition for backward induction presented by Aumann is *common knowledge of rationality.* Let ω be a state, i a player and h_i an information set controlled by i. At state ω, player i is said to be rational at information set h_i if there is no $s_i \in S_i$ such that for every $\omega' \in B_i(\omega)$ it holds that

$$v_i(\omega)(z(s_i, (f_j(\omega'))_{j\neq i}|h_i)) > v_i(\omega)(z(f_i(\omega), (f_j(\omega'))_{j\neq i}|h_i)),$$

where $z(s_i, (f_j(\omega'))_{j\neq i}|h_i)$ is the terminal node that is reached if the game would start at h_i, and the players would choose in accordance with $(s_i, (f_j(\omega'))_{j\neq i})$. In terms of our base model, this means that strategy $f_i(\omega)$ is rational for player i at h_i with respect to the utility function $v_i(\omega)$ and his first-order belief $\{(f_j(\omega'))_{j\neq i} \mid \omega' \in B_i(\omega)\}$ about the opponents' choices after h_i. Let Ω^{rat} be the set of states ω such that at ω all players are rational at each of their information sets.

Common knowledge of rationality can now be defined by the following recursive procedure:

$$\begin{aligned} \text{CKR}^1 &= \Omega^{\text{rat}}; \\ \text{CKR}^{k+1} &= \{\omega \in \Omega \mid B_i(\omega) \subseteq \text{CKR}^k \text{ for all players } i\} \end{aligned}$$

for $k \geq 1$. Then, common knowledge of rationality is said to hold at ω if $\omega \in \text{CKR}^k$ for all k. In Theorem A, Aumann proves that for every profile $(\tilde{v}_j)_{j\in I}$ of utility functions, for every state ω at which common knowledge of $(\tilde{v}_j)_{j\in I}$ and common knowledge of rationality hold, and for every player i, the strategy $f_i(\omega)$ is the backward induction strategy for player i with respect to $(\tilde{v}_j)_{j\in I}$. In Theorem B, Aumann proves that there is an Aumann model and a state ω at which common knowledge of $(\tilde{v}_j)_{j\in I}$ and common knowledge of rationality hold.

In terms of our base model, common knowledge of rationality implies common initial belief in rationality at all information sets. By the latter we mean that a type (1) initially believes that all players choose rationally at all information sets, (2) initially believes that every type initially believes that all players choose rationally at all information sets, and so on. Together with the "no substantial belief revision condition" above, this would imply that a type *always* believes that types initially believe that all players choose rationally at all information sets, and that a type always believes that types always believe that types initially believe that players choose rationally at all information sets, and so on. Hence, a possible interpretation of Aumann's condition of common knowledge of rationality, together with the "no substantial belief revision condition", in our base model would be: common belief in the event that players initially believe in rationality at all information sets. Similarly, common knowledge of $(\tilde{v}_j)_{j \in I}$, together with the "no substantial belief revision condition", could be interpreted as common belief in the event that types have preferences according to $(\tilde{P}_j)_{j \in I}$, where $\tilde{P}_j$ is the preference relation that corresponds to $\tilde{v}_j$. That is, Aumann's sufficient conditions for backward induction could be interpreted as follows in terms of our base model:

Aumann's condition BR: Type t_i respects common belief in the events that (1) types hold preferences as specified by $(\tilde{P}_j)_{j \in I}$, (2) types initially believe in rationality at all information sets, and (3) types never revise their beliefs about the opponents' choices and beliefs at future and parallel information sets.

In [Cl$_0$03], Clausing basically provides a reformulation of Aumann's model and definitions in a syntactic framework. Clausing's sufficient condition for backward induction is a little weaker than Aumann's, since Clausing only requires "true $(k-1)$st level belief" in rationality at all information sets, where k is the maximal length of a path in the game tree, which is weaker than common knowledge of rationality as defined by Aumann. Quesada proves, in [Qu03, Propositions 3.3 & 3.4] that Aumann's backward induction theorem can also be shown without imposing that $\omega \in B_i(\omega)$ for all ω, and without imposing that for all $\omega, \omega' \in \Omega$ it must hold that $B_i(\omega)$ and $B_i(\omega')$ are either identical, or have an empty intersection. That is, Quesada no longer assumes a partition structure, nor does he require that what one believes must be true. The only substantial conditions that [Qu03] imposes on the belief operator is that $f_i(\omega') = f_i(\omega)$ whenever $\omega' \in B_i(\omega)$, and similarly for v_i. Hence, a player must be aware of his choice and his utility function.

However, since the models by Clausing and Quesada [Cl$_0$03, Qu03] are identical in spirit to Aumann's, we omit a formal discussion of these models in this overview.

3.4 Balkenborg & Winter's model

In [Ba$_3$Wi$_2$97], Balkenborg and Winter present a state-based semantic model that is almost identical to Aumann's model, so we do not repeat it here.

The only difference is that Balkenborg and Winter restrict attention to extensive form structures in which every player controls only one information set. However, the sufficient conditions given for backward induction are different from Aumann's conditions, as they are based on the notion of *forward knowledge of rationality* rather than common knowledge of rationality.

For every player i, let h_i be the unique information set controlled by player i. The definition of player i being rational at h_i is the same as in Aumann's model. Let Ω_i^{rat} be the set of states ω such that at ω, player i is rational at h_i. We say that player j comes after player i if h_j comes after h_i. Forward knowledge of rationality can now be defined by the following recursive procedure. For every player i define:

$$\begin{array}{lcl} \text{FKR}_i^1 & = & \Omega_i^{\text{rat}}; \\ \text{FKR}_i^{k+1} & = & \{\omega \in \Omega \mid B_i(\omega) \subseteq \text{FKR}_j^k \text{ for all } j \text{ that come after } i\}, \end{array}$$

for every $k \geq 1$. Then, forward knowledge of rationality is said to hold at state ω if $\omega \in \text{FKR}_i^k$ for all i and all k. That is, player i believes that every player after him will choose rationally, believes that every player after him believes that every player after him will choose rationally, and so on. So, it corresponds to our notion of forward belief in substantive rationality in Definition 2.6.

In Theorem 2.1, Balkenborg and Winter prove that for every profile $(\tilde{v}_j)_{j\in I}$ of utility functions, for every state ω at which common knowledge of $(\tilde{v}_j)_{j\in I}$ and forward knowledge of rationality hold, and for every player i, the strategy $f_i(\omega)$ is the backward induction strategy for player i with respect to $(\tilde{v}_j)_{j\in I}$. Balkenborg and Winter's sufficient condition for backward induction may thus be phrased as follows in terms of our base model:

Balkenborg & Winter's condition BR: Type t_i (1) respects common belief in the event that types hold preferences as specified by $(\tilde{P}_j)_{j\in I}$, (2) respects forward belief in substantive rationality, and (3) respects common belief in the event that types never revise their beliefs about the opponents' choices and beliefs at future and parallel information sets.

Quesada proves in [Qu03, Proposition 3.1] that Balkenborg and Winter's sufficient condition for backward induction would still be sufficient if one weakens the conditions on the knowledge operators as explained at the end of the previous subsection.

3.5 Clausing's model

Clausing presents a syntactic model for games with perfect information [Cl$_0$04]. For our purposes here it is not necessary to discuss the complete formalism of Clausing's model, and therefore we restrict ourselves to presenting only the key ingredients. A major technical difference between our description here and Clausing's original model is that we shall employ "statements" instead of logical propositions. The reader is referred to the original paper for the syntactic formalism employed by Clausing. For our restricted purposes here, a Clausing model may be described as a tuple

$$\mathcal{M} = (L, (\hat{B}_i, v_i)_{i \in I})$$

where L is a language, or set of statements, $\hat{B}_i$ is a function that assigns to every statement $f \in L$ some subset $\hat{B}_i(f) \subseteq L$ of statements, and v_i is a utility function for player i on the set of terminal nodes. By "$g \in \hat{B}_i(f)$" we mean the statement that "player i believes statement g upon learning that f holds". It is assumed that L contains all statements of the form "player i chooses strategy s_i", and that it is closed under the operations $\neg$ (not), $\wedge$ (and) and $\hat{B}_i$. By the latter, we mean that if f and g are statements in L, then so are the statements "$\neg f$", "$f \wedge g$" and "$g \in \hat{B}_i(f)$".

Clausing's sufficient condition for backward induction is *forward belief from the root to all information sets h in rationality at h*. We say that strategy s_i is rational for player i at information set h_i if there is no other strategy $s_i' \in S_i(h_i)$ such that player i would believe, upon learning that h_i has been reached, that s_i' would lead to a higher utility than s_i. Formally, there should be no $s_i' \in S_i(h_i)$ and no statement $f \in L$ about the opponents' strategy choices such that (1) player i believes f upon learning that all opponents j have chosen a strategy in $S_j(h_i)$, and (2) for every opponents' strategy profile s_{-i} compatible with f it would be true that $v_i(z(s_i', s_{-i}|h_i)) > v_i(z(s_i, s_{-i}|h_i))$. Player i is said to believe at h_i that player j is rational at h_j if, upon learning that h_i has been reached, player i believes the statement "player j chooses a strategy that is rational for j at h_j". Forward belief from the root to all information sets h in rationality at h can now be defined by the following sequence of statements:

$\mathrm{FB}_i^1(h_i) =$ "player i believes, upon learning that h_i has been reached, that every opponent j will be rational at all h_j that follow h_i"

for all players i and all $h_i \in H_i^*$, and

$\mathrm{FB}_i^{k+1}(h_i) =$ "player i believes, upon learning that h_i has been reached, the statement $\mathrm{FB}_j^k(h_j)$ for all opponents j and all h_j that follow h_i"

for all players i, $h_i \in H_i^*$ and $k \geq 1$. Player i is said to respect forward belief from the root to all information sets h in rationality at h if for every h_i, player i believes, upon learning that h_i has been reached, the statements $\mathrm{FB}_j^k(h_j)$ for all k, all opponents j and all $h_j \in H_j$ that follow h_i. In Proposition 2, Clausing shows that this condition implies backward induction, whereas his Proposition 3 demonstrates that this condition is possible. In terms of our base model, Clausing's condition clearly corresponds to forward belief in substantive rationality.

Clausing's condition BR: Type t_i (1) respects common belief in the event that types hold preferences as specified by $(\tilde{P}_j)_{j\in I}$, and (2) respects forward belief in substantive rationality.

3.6 Feinberg's model

Feinberg provides a syntactic model for dynamic games which is similar to Clausing's model [Fe$_1$05]. Since a full treatment of Feinberg's model would take us too far afield, we present a highly condensed version of his model here, which will serve for our restricted purposes. As with the discussion of Clausing's model, we refer to the original paper for the syntactic formalism. For our purposes here, a Feinberg model may be described as a tuple

$$\mathcal{M} = (L, (C_i, v_i)_{i\in I})$$

where L is a language, or set of statements, C_i is a function that selects for every information set $h_i \in H_i^*$ a set $C_i(h_i) \subseteq L$ of statements, and v_i is a utility function for player i on the set of terminal nodes. The interpretation of $f \in C_i(h_i)$ is that player i is *confident* of statement f at information set h_i. The language L must contain all statements of the form "player i chooses strategy s_i", and must be closed under the application of the operators $\neg$ (not), $\wedge$ (and) and $C_i(h_i)$. By the latter we mean that, if f is a statement in L, then the statement "$f \in C_i(h_i)$" must also be in L.

Feinberg characterizes the *confidence operator* by means of a list of axioms, which largely coincides with the list of classic axioms for a knowledge operator. The single, but crucial, difference is that a player may be confident of a statement that is objectively wrong, whereas this is not possible in the case of a knowledge operator. However, in Feinberg's model a player must always be confident that he is right, that is, a player must be confident that all statements he is confident of are true.

Feinberg presents two different sufficient conditions for backward induction, namely *common confidence of hypothetical rationality* and *iterated future confidence of rationality.* Strategy s_i is said to be rational for player i at h_i if there is no other strategy $s_i' \in S_i(h_i)$ such that player i would be confident at h_i that s_i' would lead to a higher utility than s_i. By the latter, we mean that there should be no $s_i' \in S_i(h_i)$, and no statement f about the opponents' strategy choices, such that (1) i is confident of f at h_i, and (2) for every opponents' strategy profile s_{-i} compatible with f it would hold that $v_i(z(s_i', s_{-i})|h_i) > v_i(z(s_i, s_{-i})|h_i)$. We say that i is confident at h_i that j is rational at h_j if the statement "player j chooses a strategy that is rational for j at h_j" belongs to $C_i(h_i)$. Common confidence in hypothetical rationality can now be defined recursively by the following sequence of statements:

$$\text{CCHR}^1 = \text{“every player } i \text{ is confident at every } h_i \text{ that every opponent } j \text{ will be rational at every } h_j \text{ not preceding } h_i\text{”}$$

and, for every $k \geq 1$,

$$\text{CCHR}^{k+1} = \text{“every player } i \text{ is confident at every } h_i \text{ of CCHR}^k\text{”}.$$

Player i is said to respect common confidence in hypothetical rationality if, for every h_i and every k, player i is confident at h_i of CCHR^k. In Proposition 10, Feinberg shows that this condition is possible, and implies backward induction. In terms of our base model, this condition corresponds exactly to our definition of common belief in the event that types always believe in rationality at all future and parallel information sets.

Feinberg's first condition BR: Type t_i respects common belief in the events that (1) types hold preference relations as specified by $(\tilde{P}_j)_{j \in I}$, and (2) types always believe in rationality at all future and parallel information sets.

Iterated future confidence of rationality can be defined by means of the following sequence of statements:

$$\text{IFCR}_i^1(h_i) = \text{“player } i \text{ is confident at } h_i \text{ that all opponents } j \text{ will be rational at all } h_j \text{ that follow } h_i\text{”}$$

for all $i \in I$ and all $h_i \in H_i^*$, and

$$\text{IFCR}_i^{k+1}(h_i) = \text{“player } i \text{ is confident at } h_i \text{ of IFCR}_j^k(h_j) \text{ for all opponents } j \text{ and all } h_j \text{ that follow } h_i\text{”}$$

for all $i \in I$, $h_i \in H_i^*$ and $k \geq 1$. Player i is said to respect iterated future confidence of rationality if, for every k, every h_i, every opponent j, and every h_j following h_i, player i is confident at h_i of IFCR$_j^k(h_j)$. Feinberg shows in his Proposition 11 that this condition is possible and leads to backward induction. In terms of our base model, this condition corresponds to our definition of forward belief in substantive rationality.

Feinberg's second condition BR: Type t_i (1) respects common belief in the event that types hold preferences as specified by $(\tilde{P}_j)_{j \in I}$, and (2) respects forward belief in substantive rationality.

3.7 Perea's model

Perea proposes a type-based semantic model that is very similar to our base model [Pe$_2$05]. The difference is that in [Pe$_2$05], the players' initial and revised beliefs are assumed to be point-beliefs, that is, contain exactly one strategy-type pair for each opponent. Moreover, in [Pe$_2$05] the model is assumed to be *complete* which will be defined below. An important difference between Perea's model and the other models discussed here is that Perea's model explicitly allows for the possibility that players revise their belief about the opponents' preference relations over terminal nodes as the game proceeds. A Perea model is a tuple

$$\mathcal{M} = (T_i, P_i, \hat{B}_i)_{i \in I}$$

where T_i is player i's set of types, P_i assigns to every type $t_i \in T_i$ a strict preference relation $P_i(t_i)$ over the terminal nodes, $\hat{B}_i$ assigns to every type $t_i \in T_i$ and every information set $h_i \in H_i^*$ a belief $\hat{B}_i(t_i, h_i) \subseteq \prod_{j \neq i}(S_j(h_i) \times T_j)$ consisting of exactly one point, and the model $\mathcal{M}$ is *complete.* The assumption that the belief $\hat{B}_i(t_i, h_i)$ consists of exactly one point means that t_i, at every information set h_i, is supposed to consider only one strategy-type pair (s_j, t_j) possible for every opponent j. However, this point-belief may change as the game proceeds. By a complete model, we mean that for every player i, every strict preference relation $\hat{P}_i$ and every belief vector $\tilde{B}_i = (\tilde{B}_i(h_i))_{h_i \in H_i^*}$ consisting of conditional point-beliefs $\tilde{B}_i(h_i)$ as described above, there is some type $t_i \in T_i$ with $P_i(t_i) = \hat{P}_i$ and $\hat{B}_i(t_i, h_i) = \tilde{B}_i(h_i)$ for all h_i. Since types may revise their belief about the opponents' types, and different types may have different preference relations over terminal nodes, Perea's model allows types to revise their belief about the opponents' preference relations over terminal nodes.

Perea's sufficient condition for backward induction is common belief in the events that (1) players initially believe in $(\tilde{P}_i)_{i \in I}$, (2) players initially believe in rationality at all information sets, and (3) the players' belief revision procedures satisfy some form of minimal belief revision. The crucial difference with the other models discussed here is that condition (1) allows players to revise their belief about the opponents' preference relations as the game proceeds. On the other hand, the conditions (2) and (3) as they have been defined in [Pe$_2$05] can be shown to imply that players should *always* believe that every opponent chooses rationally at *all information sets;* a condition that cannot be realized in general if players do not revise their beliefs about the opponents' preference relations.

A type t_i is said to initially believe in $(\tilde{P}_j)_{j \in I}$ if for every opponent j, the initial belief $\hat{B}_i(t_i, h_0)$ about player j consists of a strategy-type pair (s_j, t_j) where $P_j(t_j) = \tilde{P}_j$. In order to formalize condition (3), we need the definition of an elementary statement. A first-order elementary statement about player i is a statement of the form "player i has a certain preference relation" or "player i believes at h_i that opponent j chooses a certain strategy". Recursively, one can define, for every $k \geq 2$, a kth order elementary statement about player i as a statement of the form "player i believes at h_i that φ" where φ is a $(k-1)$st order elementary statement. An elementary statement about player i is then an elementary statement about player i of some order k. Now, let $h_i \in H_i \backslash h_0$, and let h_i' be the information set in H_i^* that precedes h_i and for which no other player i information set is between h_i' and h_i. For every opponent j, let (s_j', t_j') be the strategy-type pair in $\hat{B}_i(t_i, h_i')$, and let (s_j, t_j) be the strategy-type pair in $\hat{B}_i(t_i, h_i)$. Type t_i is said to satisfy minimal belief revision at h_i if for every opponent j the strategy-type pair (s_j, t_j) is such that (1) s_j is rational for t_j at all information sets, (2) there is no other strategy-type pair (s_j'', t_j'') in $S_j(h_i) \times T_j$ satisfying (1) such that t_j'' and t_j' disagree on fewer elementary statements about player j than t_j and t_j' do, and (3) there is no other strategy-type pair (s_j'', t_j'') in $S_j(h_i) \times T_j$ satisfying (1) and (2) such $P_j(t_j'')$ and $P_j(t_j')$ disagree on fewer pairwise rankings of terminal nodes than $P_j(t_j)$ and $P_j(t_j')$ do. It can be shown that this notion of minimal belief revision, together with the condition that players initially believe in rationality at all information sets, imply that a type always believes that his opponents choose rationally at *all* information sets. For the definition of minimal belief revision it is very important that the model $\mathcal{M}$ is assumed to complete. [Pe$_2$05, Theorem 5.1] shows that there is a Perea model which satisfies the sufficient condition listed above. Theorem 5.2 in that paper demonstrates that this sufficient condition leads to backward induction. As such, Perea's sufficient condition for backward induction can be stated as follows in terms of our base model:

Perea's condition BR: Type t_i respects common belief in the events that (1) types hold point-beliefs, (2) types initially believe in $(\tilde{P}_j)_{j \in I}$, (3) types always believe in rationality at all information sets, and (4) types satisfy minimal belief revision.

3.8 Quesada's model

Quesada presents a model for games with perfect information which is neither semantic nor syntactic [Qu02]. The key ingredient is to model the players' uncertainty by means of *Bonanno belief systems* [Bo$_0$92]. A Bonnano belief system is a profile $\beta = (\beta_i)_{i \in I}$, where β_i is a belief vector that assigns to every information set h (not necessarily controlled by player i) some terminal node $\beta_i(h)$ which follows h. The interpretation is that player i, upon learning that the game has reached information set h, believes that he and his opponents will act in such a way that terminal node $\beta_i(h)$ will be reached. A Quesada model is a pair

$$\mathcal{M} = (\mathcal{B}, (v_i)_{i \in I})$$

where $\mathcal{B}$ is a set of Bonnano-belief systems, and v_i is a utility function for player i over the terminal nodes. Quesada's sufficient condition for backward induction states that every belief system in $\mathcal{B}$ should be *rational*, and that every belief system in $\mathcal{B}$ should be *justifiable* by other belief systems in $\mathcal{B}$. Formally, a belief system $\beta = (\beta_i)_{i \in I}$ is said to be rational if for every player i and every information set $h_i \in H_i$ it holds that $v_i(\beta_i(h_i)) \geq v_i(\beta_i((h_i, a)))$ for every action $a \in A(h_i)$, where (h_i, a) denotes the information set that immediately follows action a at h_i. We say that belief system $\beta = (\beta_i)_{i \in I}$ in $\mathcal{B}$ is justifiable by other belief systems in $\mathcal{B}$ if for every player i, every $h_i \in H_i$, every opponent j, and every $h_j \in H_j$ between h_i and the terminal node $\beta_i(h_i)$ there is some belief system $\beta' = (\beta'_i)_{i \in I}$ in $\mathcal{B}$ such that $\beta'_j(h_j) = \beta_i(h_i)$. A belief system $\beta = (\beta_i)_{i \in I}$ is called the backward induction belief system if for every player i and every information set h, $\beta_i(h)$ is the terminal node which is reached by applying the backward induction procedure (with respect to $(v_i)_{i \in I}$) from h onwards. In Proposition 1, Quesada shows that there is one, and only one, set $\mathcal{B}$ which satisfies the two conditions above, namely the set containing only the backward induction belief system.

We shall now attempt to express these conditions in terms of our base model. Take a set $\mathcal{B}$ of belief systems such that every belief system in $\mathcal{B}$ is justifiable by other belief systems in $\mathcal{B}$ (and thus satisfies Quesada's second condition above). Then, every belief system β_i in $\mathcal{B}$ induces, for every h_i, a point-belief about the opponents' strategy choices as follows: For every h_i there is some opponents' strategy profile $s_{-i}(\beta_i, h_i) \in \prod_{j \neq i} S_j(h_i)$ such that, for every action $a \in A(h_i)$, the action a followed by $s_{-i}(\beta_i, h_i)$ leads to the terminal node $\beta_i(h_i, a)$. Hence, $s_{-i}(\beta_i, h_i)$ may be interpreted as β_i's conditional point-belief at h_i about the opponents' strategy choices. (Note that this belief need not be unique, as β_i does not restrict player i's beliefs at h_i about opponents' choices at parallel information sets). The belief vector β_i also induces, for every h_i, a conditional point-belief about the opponents' belief vectors β'_j in $\mathcal{B}$. Consider, namely, an information set $h_i \in H_i$, some opponent j and an information set h_j between h_i and the terminal node $\beta_i(h_i)$ such that there is no further player j information set between h_i and h_j. Since $\mathcal{B}$ satisfies Quesada's justifiability condition, there is some player j belief vector $\beta_j(\beta_i, h_i)$ in $\mathcal{B}$ such that $\beta_j(\beta_i, h_i)(h_j) = \beta_i(h_i)$. (Again, this choice need not be unique). This belief vector $\beta_j(\beta_i, h_i)$ may then serve as β_i's conditional point-belief at h_i about player j's belief vector. Summarizing, every belief vector β_i induces, at every h_i, a conditional point-belief about the opponents' strategy choices and the opponents' belief vectors.

Now, if we interpret every belief vector β_i in $\mathcal{B}$ as a type $t_i(\beta_i)$ in our base model, then, by the insights above, every type $t_i(\beta_i)$ induces, at every h_i, a conditional point-belief about the opponents' strategy choices and types $t_j(\beta_j)$. Hence, similarly to Perea's model, Quesada's model can be expressed in terms of our base model by imposing common belief in the event that types hold point-beliefs. Let $T_i(\mathcal{B})$ denote the set of all such types

$t_i(\beta_i)$ induced by some belief vector β_i in $\mathcal{B}$. A combination of Quesada's rationality condition and justifiability condition implies that, whenever β_i in $\mathcal{B}$ believes at h_i that player j chooses action a at some h_j between h_i and $\beta_i(h_i)$ (with no player j information set between h_i and h_j), then there is some rational belief vector $\beta_j(\beta_i, h_i)$ in $\mathcal{B}$ such that $\beta_j(\beta_i, h_i)(h_j) = \beta_i(h_i)$. In particular, action a must be part of the rational belief vector $\beta_j(\beta_i, h_i)$, and hence action a must be optimal with respect to $\beta_j(\beta_i, h_i)$. In terms of our base model, this means that, whenever type $t_i(\beta_i)$ believes at h_i that information set h_j will be reached in the future, and believes at h_i that player j will choose action a at h_j, then $t_i(\beta_i)$ must believe at h_i that player j is of some type $t_j(\beta_j)$ for which a is rational. In other words, every type $t_i(\beta_i)$ in $T_i(\mathcal{B})$ always believes in rationality at future information sets that are believed to be reached. However, since $t_i(\beta_i)$ believes at every information set that every opponent j is of some type $t_j(\beta_j)$ in $T_j(\mathcal{B})$, it follows that every $t_i(\beta_i)$ in $T_i(\mathcal{B})$ always believes in the event that all types believe in rationality at future information sets that are believed to be reached. By recursively applying this argument, one may conclude that every $t_i(\beta_i)$ in $T_i(\mathcal{B})$ respects common belief in the event that types always believe in rationality at future information sets that are believed to be reached. Quesada's sufficient condition can thus be formulated as follows in terms of our base model:

Quesada's condition BR: Type t_i respects common belief in the events that (1) types hold preferences as specified by $(\tilde{P}_j)_{j\in I}$, (2) types hold point-beliefs, and (3) types always believe in rationality at future information sets that are believed to be reached.

3.9 Samet's model

Samet presents a state-based semantic model which is an extension of the models by Aumann and Balkenborg & Winter [Sa$_2$96]. A Samet model is a tuple

$$\mathcal{M} = (\Omega, (B_i, f_i, v_i, \tau_i)_{i\in I}),$$

where Ω, B_i, f_i and v_i are as in the Aumann model, and τ_i is a so-called *hypothesis transformation* that assigns to every state ω and non-empty event $E \subseteq \Omega$ some new state ω'. My interpretation of τ_i is that if player i currently believes that the state is in $B_i(\omega)$, but later observes the event E, then he will believe that the state is in $B_i(\omega') \cap E$. Samet defines the hypothesis transformation in a different, but equivalent, way. In Samet's terminology, a hypothesis transformation assigns to every initial belief $B_i(\omega)$ and event E some new belief $B_i(\omega')$ for some $\omega' \in \Omega$. However, this definition is equivalent to the existence of a function τ_i as described in our model. The function τ_i must satisfy the following two conditions: (1) $B_i(\tau_i(\omega, E)) \cap E$ is nonempty for every ω and E, and (2) $\tau_i(\omega, E) = \omega$ whenever $B_i(\omega)$ has a nonempty intersection with E. These conditions indicate that $B_i(\tau_i(\omega, E)) \cap E$ may be interpreted as a well-defined conditional belief for player i at state ω when observing the event E.

As to the functions f_i, mapping states to strategy choices, it is assumed that for every terminal node z there is some state $\omega \in \Omega$ such that the profile $(f_i(\omega))_{i \in I}$ of stategies reaches z. This implies that for every information set h_i, the event

$$[h_i] = \{\omega \in \Omega \mid (f_i(\omega))_{i \in I} \text{ reaches } h_i\}$$

is nonempty, and hence can be used as a conditioning event for the hypothesis transformation τ_i. Samet assumes in his model a function ξ (instead of $(f_i)_{i \in I}$) mapping states to terminal nodes, and assumes that for every terminal node z there is some $\omega \in \Omega$ with $\xi(\omega) = z$. However, he shows that this function ξ induces, in some precise way, a profile $(f_i)_{i \in I}$ of strategy functions, as we use it. We work directly with the strategy functions here, in order to make the model as similar as possible to the Aumann model and the Balkenborg-Winter model.

In contrast to Aumann's model and Balkenborg and Winter's model, every state ω in Samet's model formally induces a conditional belief vector in our base model. Namely, take some state ω, a player i, and some information set $h_i \in H_i^*$. Then,

$$\hat{B}_i(\omega, h_i) := B_i(\tau_i(\omega, [h_i])) \cap [h_i]$$

respresents player i's conditional belief at h_i about the state. Since every state ω' induces for player j a strategy choice $f_j(\omega')$ and a conditional belief vector $(\hat{B}_j(\omega', h_j))_{h_j \in H_j^*}$, first-order conditional beliefs about the opponents' strategies, and higher-order conditional beliefs about the opponents' conditional beliefs can be derived at every state with the help of the hypothesis transformations τ_i. Hence, Samet's model can be expressed directly and formally in terms of our base model.

Samet's sufficient condition for backward induction is *common hypothesis of node rationality.* At state ω, player i said to be rational at $h_i \in H_i$ if (1) $\omega \in [h_i]$, and (2) there is no $s_i \in S_i$ such that for every $\omega' \in B_i(\omega) \cap [h_i]$ it holds that

$$v_i(\omega)(z(s_i, (f_j(\omega'))_{j \neq i} | h_i)) > v_i(\omega)(z(f_i(\omega), (f_j(\omega'))_{j \neq i} | h_i)),$$

where the definition of this expression is as in Aumann's model. Let $[\mathrm{rat}_i(h_i)]$ denote the set of states ω such that at ω, player i is rational at h_i. Common hypothesis of node rationality can now be defined by the following recursive procedure: For every player i and information set $h_i \in H_i^*$, let

$$\mathrm{CHNR}(h_i, h_i) = [\mathrm{rat}_i(h_i)].$$

Note that, by condition (1) above, $\mathrm{CHNR}(h_i, h_i)$ only contains states at which h_i is indeed reached. Now, let $k \geq 0$, and suppose that $\mathrm{CHNR}(h_i, h_j)$ has been defined for all information sets $h_i \in H_i^*, h_j \in H_j^*$ such that h_j comes after h_i, and there are at most k information sets between h_i and h_j. Suppose now that h_j comes after h_i, and that there are exactly $k+1$ information sets between h_i and h_j. Let h be the unique information set that immediately follows h_i and precedes h_j. Define

$$\mathrm{CHNR}(h_i, h_j) = \{\omega \in \Omega \mid B_i(\tau_i(\omega, [h])) \cap [h] \subseteq \mathrm{CHNR}(h, h_j)\}.$$

Common hypothesis of node rationality is said to hold at state ω if $\omega \in \mathrm{CHNR}(h_0, h)$ for all information sets h. Hence, the player at h_0 believes that (1) every opponent j will choose rationally at those information sets h_j that immediately follow h_0, (2) every such opponent j will believe at every such h_j that every other player k will choose rationally at those h_k that immediately follow h_j, and so on.

Samet shows in Theorem 5.3 that for every profile $(v_i)_{i \in I}$ of utility functions, for every state ω at which common knowledge of $(v_i)_{i \in I}$ and common hypothesis of node rationality hold, the strategy profile $(f_i(\omega))_{i \in I}$ leads to the backward induction outcome with respect to $(v_i)_{i \in I}$. In particular, the player at h_0 chooses the backward induction action at h_0 with respect to $(v_i)_{i \in I}$. In Theorem 5.4, Samet shows that there always exists some state ω at which common knowledge of $(v_i)_{i \in I}$ and common hypothesis of node rationality hold.

For a given state ω and information set $h_i \in H_i^*$, say that common hypothesis of node rationality *at* h_i holds if $\omega \in \mathrm{CHNR}(h_i, h)$ for all information sets h that follow h_i. Then, Samet's Theorem 5.3 can be generalized as follows: For every $h_i \in H_i$ and every ω at which common knowledge of $(v_i)_{i \in I}$ and common hypothesis of node rationality at h_i hold, the strategy $f_i(\omega)$ chooses at h_i the backward induction action with respect to h_i.

In order to express this sufficient condition in terms of our base model, it is important to understand all implications of common hypothesis of node rationality. By definition, common hypothesis of node rationality at h_i implies that player i believes at h_i that (1) every opponent j will choose rationally at every information set h_j that immediately follows h_i, (2) every such opponent j will believe at every such h_j that every other player k will choose rationally at every h_k that immediately follows h_j, and so on. However, there are more implications.

Consider namely an information set $h_j \in H_j$ that immediately follows h_i and some information set $h_k \in H_k$ which immediately follows h_j such that $B_i(\tau_i(\omega, [h_j])) \subseteq [h_k]$. Hence, in terms of our base model, player i believes at h_i that h_k will be reached. Suppose that state ω is such that common hypothesis of node rationality at h_i holds at ω. By (1) above, it holds that (1') $B_i(\tau_i(\omega,[h_j]))\cap[h_j]\subseteq[\mathrm{rat}_j(h_j)]$. By (2) above, it holds for every $\omega'\in B_i(\tau_i(\omega,[h_j]))\cap[h_j]$ that (2') $B_j(\tau_j(\omega',[h_k]))\cap[h_k]\subseteq[\mathrm{rat}_k(h_k)]$. However, since $B_i(\tau_i(\omega,[h_j]))\subseteq[h_k]$, it follows that $\omega'\in[h_k]$ for every $\omega'\in B_i(\tau_i(\omega,[h_j]))$. Since $\omega'\in B_j(\omega')$, we have that $B_j(\omega')$ has a nonempty intersection with $[h_k]$, and hence (by the assumptions on τ_i) $\tau_j(\omega',[h_k])=\omega'$ for every $\omega'\in B_i(\tau_i(\omega,[h_j]))$. We may therefore conclude that $B_j(\tau_j(\omega',[h_k]))=B_j(\omega')$ for every $\omega'\in B_i(\tau_i(\omega,[h_j]))\cap[h_j]$. By (2') it thus follows that $B_j(\omega')\cap[h_k]\subseteq[\mathrm{rat}_k(h_k)]$ for every $\omega'\in B_i(\tau_i(\omega,[h_j]))\cap[h_j]$. Since $\omega'\in B_j(\omega')$, and $\omega'\in[h_k]$ for every $\omega'\in B_i(\tau_i(\omega,[h_j]))$, it follows in particular that $\omega'\in[\mathrm{rat}_k(h_k)]$ for every $\omega'\in B_i(\tau_i(\omega,[h_j]))\cap[h_j]$, which means that player i believes at h_i that player k chooses rationally at h_k. Hence, we have shown that common hypothesis of node rationality at h_i implies that player i believes at h_i that player k chooses rationally at h_k whenever (1) there is only one information set between h_i and h_k, and (2) player i believes at h_i that h_k will be reached. By induction, one can now show that common hypothesis of node rationality at h_i implies that player i believes at h_i that player k chooses rationally at h_k whenever (1) h_k follows h_i and (2) player i believes at h_i that h_k can be reached.

By a similar argument, one can show that common hypothesis of node rationality at h_i implies that player i believes at h_i that common hypothesis of node rationality will hold at every future information set h_j which player i believes to be reached from h_i. Together with our previous insight, this means that common hypothesis of node rationality may be expressed, in terms of our base model, by forward belief in material rationality (see our Definition 2.7). Samet's sufficient condition for backward induction can thus be phrased as follows in terms of our base model:

Samet's condition BR: Type t_i (1) respects common belief in the event that types hold preferences as specified by $(\tilde{P}_j)_{j\in I}$, and (2) respects forward belief in material rationality.

3.10 Stalnaker's model

Stalnaker proposes a state-based semantic model for perfect information games in which every information set is controlled by a different player [St$_1$98]. The model we present here is not an exact copy of Stalnaker's model, but captures its essential properties. A Stalnaker model is a tuple

$$\mathcal{M}=(\Omega,(f_i,v_i,\lambda_i)_{i\in I})$$

where Ω, f_i and v_i are as in the Aumann model, and λ_i is a function that assigns to every state ω some lexicographic probability system (see Asheim's model) $\lambda_i(\omega)$ on Ω. That is, $\lambda_i(\omega)$ is a sequence $(\lambda_i^1(\omega),\ldots,\lambda_i^{K_i(\omega)}(\omega))$ where $\lambda_i^k(\omega)$ is a probability distribution on Ω. For every information set h let $[h]=\{\omega\in\Omega\mid(f_i(\omega))_{i\in I}\text{ reaches }h\}$. We assume that $[h]$ is non-empty for all h, and that $\lambda_i(\omega)$ has full support on Ω. By the latter, we mean that for every $\omega' \in \Omega$ there is some $k \in \{1, \dots, K_i(\omega)\}$ such that $\lambda_i^k(\omega)$ assigns positive probability to ω'. As such, λ_i and $(f_j)_{j \neq i}$ induce, for every state ω, a probabilistic belief revision policy for player i in the following way. For every $h_i \in H_i^*$, let $k_i(\omega, h_i)$ be the first k such that $\lambda_i^k(\omega)$ assigns positive probability to $[h_i]$. Then, the probability distribution $\mu_i(\omega, h_i)$ on $[h_i]$ given by

$$\mu_i(\omega, h_i)(\omega') = \frac{\lambda_i^{k_i(\omega,h_i)}(\omega')}{\lambda_i^{k_i(\omega,h_i)}([h_i])}$$

for every $\omega' \in [h_i]$ represents player i's revised belief at ω upon observing that h_i has been reached. More generally, for every event $E \subseteq \Omega$, the probability distribution $\mu_i(\omega, E)$ on E given by

$$\mu_i(\omega, E)(\omega') = \frac{\lambda_i^{k_i(\omega,E)}(\omega')}{\lambda_i^{k_i(\omega,E)}(E)}$$

for every $\omega' \in E$ defines player i's revised belief upon receiving information E. Here, $k_i(\omega, E)$ is the first k such that $\lambda_i^k(\omega)$ assigns positive probability to E. The lexicographic probability system $\lambda_i(\omega)$ naturally induces, for every information set $h_i \in H_i^*$, the non-probabilistic conditional belief

$$\hat{B}_i(\omega, h_i) := \operatorname{supp} \mu_i(\omega, h_i),$$

and hence Stalnaker's model can be expressed directly in terms of our base model.

Stalnaker's sufficient condition for backward induction consists of *common initial belief in sequential rationality, and common belief in the event that players treat information about different players as epistemically independent.* Player i is called sequentially rational at ω if at every information set $h_i \in H_i^*$, the strategy $f_i(\omega)$ is optimal given the utility function $v_i(\omega)$ and the revised belief about the opponents' strategy choices induced by $\mu_i(\omega, h_i)$ and $(f_j)_{j \neq i}$. Let Ω^{srat} be the set of states at which all players are sequentially rational. Common initial belief in sequential rationality can be defined by the following recursive procedure:

$$\begin{aligned} \text{CIBSR}^1 &= \Omega^{\text{srat}}; \\ \text{CIBSR}^{k+1} &= \{\omega \in \Omega \mid \hat{B}_i(\omega, h_0) \subseteq \text{CIBSR}^k \text{ for all players } i\} \end{aligned}$$

for all $k \geq 1$. Common initial belief in sequential rationality is said to hold at ω if $\omega \in \text{CIBSR}^k$ for all k. We say that two states ω and ω' are indistinguishable for player i if $f_i(\omega) = f_i(\omega')$, $v_i(\omega) = v_i(\omega')$ and $\mu_i(\omega, h_i) = \mu_i(\omega', h_i)$ for all $h_i \in H_i^*$. An event E is said to be about player i if for every two states ω, ω' that are indistinguishable for player i, either both ω and ω'

are in E, or none is in E. We say that at ω player i treats information about different players as epistemically independent if for every two different opponents j and ℓ, for every event E_j about player j and every event E_ℓ about player ℓ, it holds that $\mu_i(\omega, E_j)(E_\ell) = \mu_i(\omega, \Omega \backslash E_j)(E_\ell)$ and $\mu_i(\omega, E_\ell)(E_j) = \mu_i(\omega, \Omega \backslash E_\ell)(E_j)$. In his theorem on page 43, Stalnaker shows that common initial belief in sequential rationality and common belief in the event that players treat information about different players as epistemically independent lead to backward induction.

In terms of our base model, common initial belief in sequential rationality corresponds to the condition that a type respects common initial belief in the event that types initially believe in rationality at all information sets. The epistemic independence condition cannot be translated that easily into our base model. The problem is that the base model only allows for beliefs conditional on *specific* events, namely events in which some information set is reached. On the other hand, in order to formalize the epistemic independence condition we need to condition beliefs on more general events. There is, however, an important consequence of the epistemic independence condition that can be expressed in terms of our base model, namely that the event of reaching information set h_i should not change player i's belief about the actions and beliefs of players that did not precede h_i. In order to see this, choose a player j that precedes h_i and a player ℓ that does not precede h_i. Note that the event of player j choosing the action leading to h_i is an event about player j, and that the event of player ℓ choosing a certain action and having a certain belief vector is an event about player ℓ. Hence, epistemic independence says that player i's belief about player ℓ's action and beliefs should not depend on whether player j has moved the game towards h_i or not. Moreover, it is exactly this consequence of epistemic independence that drives Stalnaker's backward induction result. In particular, if player i initially believes that player ℓ chooses rationally at his information set h_ℓ (which does not precede h_i), then player i should continue to believe so if he observes that h_i has been reached. If we drop the assumption that every player only controls one information set, the condition amounts to saying that a player should never revise his belief about the actions and beliefs at future and parallel information sets.

In terms of our base model, Stalnaker's sufficient condition for backward induction can thus be stated as follows:

Stalnaker's condition BR: Type t_i respects common belief in the events that (1) types hold preferences as specified by $(\tilde{P}_j)_{j \in I}$, and (2) types do not change their belief about the opponents' choices and beliefs at future and parallel information sets, and type t_i respects common initial belief in the event that (3) types initially believe in rationality at all information sets.

Halpern provides an explicit comparison between the models of Aumann and Stalnaker [Ha$_0$01].

	Ash	A&P	Aum	B&W	Cla	Fei1	Fei2	Per	Que	Sam	Sta
Common belief in event that types...											
...initially believe in rat. at all information sets			●								
...always believe in rat. at future information sets that are believed to be reached									●		
...always believe in rat. at all future information sets	●										
...always believe in rat. at all future and parallel information sets		●				●					
...always believe in rat. at all information sets								●			
Common initial belief in event that types...											
...initially believe in rat. at all inf. sets											●
Forward belief in...											
...substantive rationality				●	●		●				
...material rationality										●	
Common belief in event that types...											
...never revise belief about opponents' preference relations	●	●	●	●	●	●	●		●	●	●
...do not revise belief about opponents' choices and beliefs at future and parallel information sets			●	●							●
...minimally revise belief about opponents' preferences and beliefs								●			
...hold point-beliefs								●	●		

TABLE 1. Overview of sufficient conditions for backward induction

3.11 Summary

The discussion of the various models and sufficient conditions for backward induction can be summarized by Table 1.

The table shows that several sufficient conditions for backward induction, although formulated in completely different epistemic models, become equivalent once they have been expressed in terms of our base model. Note also that there is no model assuming common belief in the events that (1) types always believe in rationality at *all* information sets, and (2) types never revise their beliefs about the opponents' preferences over terminal nodes. This is no surprise, since the papers [Re$_3$92, Re$_3$93] have illustrated that these two events are in general incompatible. Perea's model maintains condition (1) and weakens condition (2), while the other models maintain condition (2) and weaken condition (1). Finally observe that all models assume (at least) initial common belief in the event that types initially believe in rationality at all information sets, plus some extra conditions on the players' belief revision procedures. If one would only assume the former, this would lead to the concept of *common certainty of rationality at the beginning of the game*, as defined in [BP97]. This concept is considerably weaker than backward induction, as it may not even lead to the backward induction outcome. Hence, additional conditions on the players' belief revision policies are needed in each model to arrive at backward induction.

References

[As02] G.B. Asheim. On the epistemic foundation for backward induction. *Mathematical Social Sciences* 44(2):121–144, 2002.

[AsPe$_2$05] G.B. Asheim & A. Perea. Sequential and quasi-perfect rationalizability in extensive games. *Games and Economic Behavior* 53(1):15–42, 2005.

[Au95] R.J. Aumann. Backward induction and common knowledge of rationality. *Games and Economic Behavior* 8(1):6–19, 1995.

[Au98] R.J. Aumann. On the centipede game. *Games and Economic Behavior* 23(1):97–105, 1998.

[Ba$_3$Wi$_2$97] D. Balkenborg & E. Winter. A necessary and sufficient epistemic condition for playing backward induction. *Journal of Mathematical Economics* 27(3):325–345, 1997.

[Ba$_7$97] P. Battigalli. On rationalizability in extensive games. *Journal of Economic Theory* 74(1):40–61, 1997.

[Ba$_7$Si$_1$02] P. Battigalli & M. Siniscalchi. Strong belief and forward induction reasoning. *Journal of Economic Theory* 106(2):356–391, 2002.

[BP97] E. Ben-Porath. Rationality, Nash equilibrium and backwards induction in perfect-information games. *Review of Economic Studies* 64(1):23–46, 1997.

[Bi$_1$87] K. Binmore. Modeling rational players, part I. *Economics and Philosophy* 3:179–214, 1987.

[Bo$_0$92] G. Bonanno. Rational beliefs in extensive games. *Theory and Decision* 33(2):153–176, 1992.

[Br$_1$Fr$_2$Ke$_1$04] A. Brandenburger, A. Friedenberg & H.J. Keisler. Admissibility in games, 2004. Forthcoming.

[Br$_3$Ra$_0$99] J. Broome & W. Rabinowicz. Backwards induction in the centipede game. *Analysis* 59(264):237–242, 1999.

[Ca$_2$00] J.W. Carroll. The backward induction argument. *Theory and Decision* 48(1):61–84, 2000.

[Cl$_0$03] T. Clausing. Doxastic conditions for backward induction. *Theory and Decision* 54(4):315–336, 2003.

[Cl$_0$04] T. Clausing. Belief revision in games of perfect information. *Economics and Philosophy* 20:89–115, 2004.

[Fe$_1$05] Y. Feinberg. Subjective reasoning—dynamic games. *Games and Economic Behavior* 52(1):54–93, 2005.

[Ha$_0$01] J.Y. Halpern. Substantive rationality and backward induction. *Games and Economic Behavior* 37(2):425–435, 2001.

[Ho$_0$Lo$_3$13] E.W. Hobson & A.E.H. Love, eds. *Proceedings of the Fifth International Congress of Mathematicians, Vol. 2.* Cambridge University Press, 1913.

[Pe$_0$84] D.G. Pearce. Rationalizable strategic behavior and the problem of perfection. *Econometrica* 52(4):1029–1050, 1984.

[Pe$_2$05] A. Perea. Minimal belief revision leads to backward induction, 2005. Unpublished.

[Pr00] G. Priest. The logic of backward inductions. *Economics and Philosophy* 16:267–285, 2000.

[Qu02] A. Quesada. Belief system foundations of backward induction. *Theory and Decision* 53(4):393–403, 2002.

[Qu03] A. Quesada. From common knowledge of rationality to backward induction. *International Game Theory Review* 5(2):127–137, 2003.

[$\text{Ra}_0$98] W. Rabinowicz. Grappling with the centipede. *Economics and Philosophy* 14:95–126, 1998.

[$\text{Re}_3$92] P.J. Reny. Rationality in extensive-form games. *Journal of Economic Perspectives* 6(4):103–118, 1992.

[$\text{Re}_3$93] P.J. Reny. Common belief and the theory of games with perfect information. *Journal of Economic Theory* 59(2):257–274, 1993.

[$\text{Ro}_3$81] R.W. Rosenthal. Games of perfect information, predatory pricing and the chain-store paradox. *Journal of Economic Theory* 25(1):92–100, 1981.

[$\text{Ru}_1$91] A. Rubinstein. Comments on the interpretation of game theory. *Econometrica* 59(4):909–924, 1991.

[$\text{Sa}_2$96] D. Samet. Hypothetical knowledge and games with perfect information. *Games and Economic Behavior* 17(2):230–251, 1996.

[$\text{St}_1$96] R. Stalnaker. Knowledge, belief and counterfactual reasoning in games. *Economics and Philosophy* 12:133–163, 1996.

[$\text{St}_1$98] R. Stalnaker. Belief revision in games: forward and backward induction. *Mathematical Social Sciences* 36(1):31–56, 1998.

[Ze13] E. Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachpiel. In [$\text{Ho}_0\text{Lo}_3$13, pp. 501–504].

Multitape Games

Brian Semmes

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam, The Netherlands
bsemmes@science.uva.nl

Abstract

Infinite games have been a valuable tool to characterize classes of real-valued functions. In this paper we review three important examples: the *Wadge*, *Backtrack*, and *Eraser games*. We then present two natural generalizations of these games: the *Basic Multitape game* and the *Multitape Eraser game* and show that both games characterize a class of functions satisfying a certain partition property.

1 Notation and background

Most of our notation and terminology is standard in descriptive set theory and can be found in [Ke$_0$95] or [Mo$_1$80]. As usual, for sets A and B, ${}^AB$ denotes the set of functions from A to B. In particular, ${}^\omega\omega$ denotes the set of functions from ω to ω, i.e. ${}^\omega\omega$ is the set of ω-length sequences of natural numbers. The notation ${}^{<\omega}A$ denotes the set of finite sequences of elements of A, so that ${}^{<\omega}\omega$ denotes the set of finite sequences of natural numbers. We use ${}^{\leq\omega}\omega$ to denote ${}^{<\omega}\omega \cup {}^\omega\omega$. For $s \in {}^{<\omega}\omega$, let $[s] := \{u \in {}^\omega\omega : s \subset u\}$. For $x \in {}^\omega\omega$ and $k \in \omega$, let $x \to k$ be the sequence x shifted by k places; in other words, $(x \to k)(n) := x(n+k)$.

We assume that the reader is familiar with the Borel hierarchy and $\mathbf{\Sigma}^0_\alpha$ and $\mathbf{\Pi}^0_\alpha$ notation. (The reader may consult [Ke$_0$95] or [Mo$_1$80] for further information.)

The *Baire Space* is the set ${}^\omega\omega$ with the topology generated by the basic open sets $\{[s] : s \in {}^{<\omega}\omega\}$. As is standard in set theory, we think of elements of ${}^\omega\omega$ as being *real numbers*. A real-valued function $f : {}^\omega\omega \to {}^\omega\omega$ is **continuous** if the preimage of every open set is open, in other words if $f^{-1}[Y] \in \mathbf{\Sigma}^0_1$ for every $Y \in \mathbf{\Sigma}^0_1$. For $A \subseteq {}^\omega\omega$ and $f : A \to {}^\omega\omega$, we say that f is continuous if the preimage of every open set is open in the relative topology of A.

We may consider more general classes (sets) of functions as follows. Define $\mathcal{F}(n,m) := \{f : A \to {}^\omega\omega$ such that for every $Y \in \mathbf{\Sigma}^0_n$, $f^{-1}[Y] = X \cap A$ for some $X \in \mathbf{\Sigma}^0_m\}$. For example, $\mathcal{F}(1,1)$ denotes the continuous functions

Johan van Benthem, Dov Gabbay, Benedikt Löwe (eds.). *Interactive Logic. Proceedings of the 7th Augustus de Morgan Workshop, London.* Texts in Logic and Games 1, Amsterdam University Press 2007, pp. 195–207.

and $\mathcal{F}(1,2)$ denotes the set of functions for which the preimage of every $\mathbf{\Sigma}^0_1$ set is a $\mathbf{\Sigma}^0_2$ set in the relative topology of A. (The latter are commonly known as *Baire Class 1.*) The following diagram illustrates the classes of functions under consideration in this paper.

$$\begin{tikzcd}
 & \mathcal{F}(2,3) & \\
\mathcal{F}(1,2) \arrow[ur, phantom, sloped, "\subseteq"] & & \mathcal{F}(3,3) \arrow[ul, phantom, sloped, "\subseteq"] \\
 & \mathcal{F}(2,2) \arrow[ul, phantom, sloped, "\subseteq"] \arrow[ur, phantom, sloped, "\subseteq"] & \\
\mathcal{F}(1,1) \arrow[ur, phantom, sloped, "\subseteq"] & &
\end{tikzcd}$$

The containments in the diagram (which are in fact proper) can easily be seen: in general, $\mathcal{F}(n{+}1,m) \subseteq \mathcal{F}(n,m)$ (this is immediate) and $\mathcal{F}(n,m) \subseteq \mathcal{F}(n{+}1,m{+}1)$.

We use the notation $\mathcal{F}_{\mathrm{T}}(n,m)$ to denote the set of *total* functions in $\mathcal{F}(n,m)$, i.e. $\mathcal{F}_{\mathrm{T}}(n,m) := \mathcal{F}(n,m) \cap \{f : {}^\omega\omega \to {}^\omega\omega\}$. For $A \subseteq {}^\omega\omega$ and $\mathbf{\Gamma}$ a boldface pointclass, a $\mathbf{\Gamma}$-partition of A is a pairwise disjoint sequence $\langle A_n : n < \omega\rangle$ such that $A_n = A'_n \cap A$ for some $A'_n \in \mathbf{\Gamma}$, and $\bigcup_{n<\omega} A_n = A$. For $\mathcal{F}$ a set of real-valued functions, we define $\mathcal{P}(\mathbf{\Gamma},\mathcal{F}) := \{f : A \to {}^\omega\omega$ such that there is a $\mathbf{\Gamma}$-partition $\langle A_n : n < \omega\rangle$ of A such that $f{\restriction}A_n \in \mathcal{F}\}$. We use the notation $\mathcal{P}_{\mathrm{T}}(\mathbf{\Gamma},\mathcal{F}) := \mathcal{P}(\mathbf{\Gamma},\mathcal{F}) \cap \{f : {}^\omega\omega \to {}^\omega\omega\}$. For example, a special case of a well-known theorem of Jayne and Rogers states that a function $f : {}^\omega\omega \to {}^\omega\omega$ is $\mathcal{F}(2,2)$ if and only if there is a $\mathbf{\Pi}^0_1$-partition $\langle A_n : n < \omega\rangle$ of ${}^\omega\omega$ such that $f{\restriction}A_n$ is continuous; in our notation, this may be written as $\mathcal{F}_{\mathrm{T}}(2,2) = \mathcal{P}_{\mathrm{T}}(\mathbf{\Pi}^0_1, \mathcal{F}(1,1))$.

As a final background note, we prove the following lemma.

Lemma 1.1. Let $X \subseteq {}^\omega\omega$ and $1 < \alpha < \omega_1$. If $X \in \mathbf{\Sigma}^0_\alpha$ then $X = \bigcup_{n\in\omega} X_n$ satisfying:

1) for each n, there exists $\beta_n < \alpha$ such that $X_n \in \mathbf{\Pi}^0_{\beta_n}$ and

2) $n \neq m \Rightarrow X_n \cap X_m = \varnothing$.

Proof. We begin by proving the lemma for $\alpha = 2$. Since $X \in \mathbf{\Sigma}^0_2$, by definition there is a sequence $X_n \in \mathbf{\Pi}^0_1$ such that $X = \bigcup_{n\in\omega} X_n$. Let $Y_0 := X_0$ and for $n > 0$, let $Y_n := X_n \setminus (X_0 \cup \cdots \cup X_{n-1})$. It follows that $\bigcup_{n\in\omega} Y_n$ and that the Y_n are pairwise disjoint. Moreover, each Y_n is the intersection of a closed set X_n and an open set $O_n := {}^\omega\omega \setminus (X_0 \cup \cdots \cup X_{n-1})$. Since we are working in the Baire Space, each open set O_n is the disjoint union of countably many basic open (clopen) sets, from which it follows that each Y_n is the disjoint union of countably many closed sets $Y_{n,k}$. Since the countable union of countable sets is countable, the sequence $Y_{n,k}$ is as desired.

For the inductive case, assume that the lemma holds for all $\alpha' < \alpha$ and let $X \in \mathbf{\Sigma}^0_\alpha$. By definition, $X = \bigcup_{n\in\omega} X_n$ with each $X_n \in \mathbf{\Pi}^0_{\beta_n}$ for some $\beta_n < \alpha$. Let $Y_0 := X_0$ and for $n > 0$, let $Y_n := X_n \setminus (X_0 \cup \cdots \cup X_{n-1})$. Each Y_n is the intersection of a $\mathbf{\Pi}^0_{\beta_n}$ set and a $\mathbf{\Sigma}^0_{\beta_n}$ set, so we may apply the inductive hypothesis and argue as before. Q.E.D.

2 Infinite games

We will consider a variety of *infinite token games* in this paper. In each token game, there is a set $A \subseteq {}^\omega\omega$ and a function $f : A \to {}^\omega\omega$. There are two players, Player I and Player II, who alternate moves for ω rounds. Player I begins by playing x_0, Player II responds with y_0, Player I responds with x_1, and so forth:

$$\begin{array}{lccccccccl}
\text{I:} & x_0 & & x_1 & & x_2 & & & & x = \langle x_n : n \in \omega\rangle \\
 & & & & & & & \cdots & & \\
\text{II:} & & y_0 & & y_1 & & y_2 & & & y = \langle y_n : n \in \omega\rangle
\end{array}$$

Player I plays elements $x_i \in \omega$ and Player II plays elements $y_i \in \omega \cup \{\mathsf{T}_1, \ldots, \mathsf{T}_k\}$ for some finite set of tokens $\{\mathsf{T}_1, \ldots, \mathsf{T}_k\}$. Informally, each token corresponds to an option that Player II has in the game. At the end of the game, Player I has produced a sequence $x \in {}^\omega\omega$ and Player II has produced a sequence $y \in {}^\omega(\omega \cup \{\mathsf{T}_1, \ldots, \mathsf{T}_k\})$.

The sequence y is a sequence of natural numbers and tokens—however, the rules of the game will say how to *interpret* y as a sequence (finite or infinite) of natural numbers only. Specifically, for each game we define an interpretation function $\iota : {}^\omega(\omega \cup \{\mathsf{T}_1, \ldots, \mathsf{T}_k\}) \to {}^{\leq\omega}\omega$. Player II *wins the game* if $\iota(y) = f(x)$. (If $x \notin A$ then Player II wins automatically.) Note that if $x \in A$, Player II cannot possibly win the game if $\iota(y)$ is finite.

A **strategy for Player II** is a function $\tau : {}^{<\omega}\omega \to {}^{<\omega}(\omega \cup \{\mathsf{T}_1, \ldots, \mathsf{T}_k\})$ such that $\mathrm{lh}(\tau(s)) = \mathrm{lh}(s)$ and $s \subseteq t \Rightarrow \tau(s) \subseteq \tau(t)$. The argument of τ is a finite sequence of moves by Player I and the value of τ is a finite sequence of moves by Player II. Informally, τ tells Player II what to do in the game. With respect to the above diagram, if Player II follows τ then $\tau(\langle x_0, \ldots, x_k\rangle) = \langle y_0, \ldots, y_k\rangle$ and $y = \bigcup_{s\subset x} \tau(s)$.

For a strategy τ for Player II, it is convenient to define $\hat{\tau} : {}^\omega\omega \to {}^\omega(\omega \cup \{\mathsf{T}_1, \ldots, \mathsf{T}_k\})$,

$$\hat{\tau}(x) := \bigcup_{s\subset x} \tau(s).$$

We then define $\bar{\tau}(x) := \iota(\hat{\tau}(x))$ and say that a strategy τ for Player II is **winning** if $\bar{\tau}(x) = f(x)$ for every $x \in A$.

We will proceed by defining specific token games G, each with the property that Player II has a winning strategy in $G(f)$ if and only if f belongs to some particular class of functions.

To get started, we will review the *Wadge*, *Backtrack*, and *Eraser games.*

3 The Wadge game

The *Wadge game* was developed by William Wadge in his thesis [Wa$_0$83] to characterize the notion of continuous reduction. We present a slightly modified version of the Wadge game to be consistent with our paradigm. Fix a set $A \subseteq {}^\omega\omega$ and a function $f : A \to {}^\omega\omega$. The Wadge game $G_{\mathrm{W}}(f)$ has two players, Player I and Player II, who alternate moves for ω rounds. Player I plays elements of ω and Player II plays elements of $\omega \cup \{\mathsf{P}\}$. The token P is interpreted to mean "pass".

Formally, to define the interpretation function we first define $\theta : {}^{<\omega}(\omega \cup \{\mathsf{P}\}) \to {}^{<\omega}\omega$ by $\theta(\varnothing) := \varnothing$ and

$$\theta(s^\frown\langle z\rangle) := \begin{cases} \theta(s) & \text{if } z = \mathsf{P}, \\ \theta(s)^\frown\langle z\rangle & \text{otherwise.} \end{cases}$$

We then define $\iota_{\mathrm{W}} : {}^\omega(\omega \cup \{\mathsf{P}\}) \to {}^{\leq\omega}\omega$, $\iota_{\mathrm{W}}(y) := \bigcup_{s \subset y} \theta(s)$. Letting $x \in {}^\omega\omega$ be the infinite play of Player I and $y \in {}^\omega(\omega \cup \{\mathsf{P}\})$ be the infinite play of Player II, Player II wins the game if $x \notin A$ or $\iota_{\mathrm{W}}(y) = f(x)$. (Note that in order to have a chance, Player II must play infinitely often in ω if Player I plays $x \in A$.) If τ is a Wadge strategy for Player II, we let $\hat{\tau}(x) := \bigcup_{s \subset x} \tau(s)$, $\bar{\tau}(x) := \iota_{\mathrm{W}}(\hat{\tau}(x))$ and say that τ is **winning** for Player II if $\bar{\tau}(x) = f(x)$ for all $x \in A$.

Examples. Suppose Player II plays the sequence

$$\langle a_0, a_1, \mathsf{P}, a_2, \mathsf{P}, \mathsf{P}, a_3, \dots\rangle,$$

the interpretation will be

$$\langle a_0, a_1, a_2, a_3, \dots\rangle.$$

Suppose Player II plays the sequence

$$\langle a_0, a_1, \mathsf{P}, a_2, \mathsf{P}, \mathsf{P}, \mathsf{P}, \mathsf{P}, \dots\rangle$$

with cofinally many passes, the interpretation will be

$$\langle a_0, a_1, a_2\rangle.$$

Note that Player II cannot win in this case if Player I plays in A.

Theorem 3.1 (Wadge)**.** Let $A \subseteq {}^{\omega}\omega$. A function $f : A \to {}^{\omega}\omega$ is continuous $\Leftrightarrow$ Player II has a winning strategy in the game $G_{\mathrm{W}}(f)$.

Proof. We begin by noting that

$$\bar{\tau}(x) = \bigcup_{s \subset x} \theta(\tau(s)).$$

$\Leftarrow$: Suppose τ is the winning strategy. To show that f is continuous, it suffices to show that the preimage of a basic open set is open in the topology of A. Let $t \in {}^{<\omega}\omega$ and let $X := \bigcup \{[s] : \theta(\tau(s)) = t\}$. It is not difficult to check that $f^{-1}[[t]] = X \cap A$.

$\Rightarrow$: Define τ by

$$\tau(s^\frown\langle m\rangle) := \begin{cases} \tau(s)^\frown\langle n\rangle & \text{if } f[[s^\frown\langle m\rangle]] \subseteq [\theta(\tau(s))^\frown\langle n\rangle], \\ \tau(s)^\frown\langle \mathsf{P}\rangle & \text{otherwise.} \end{cases}$$

It is not difficult to check that τ is well-defined and winning for Player II in $G_{\mathrm{W}}(f)$.

Q.E.D.

4 The Backtrack game

The Backtrack game, a generalization of the Wadge game, was developed by Robert van Wesep [VW78]. In the Backtrack game $G_{\mathrm{B}}(f)$, Player II plays elements of $\omega \cup \{\mathsf{P}, \mathsf{B}\}$. As in the Wadge game, the token P is interpreted to mean "pass." The token B, the "backtrack" option, allows Player II to erase his entire output and start playing a new sequence of natural numbers.

Let ι_{W} be defined as in the Wadge game, we define the interpretation function $\iota_{\mathrm{B}} : {}^{\omega}(\omega \cup \{\mathsf{P}, \mathsf{B}\}) \to {}^{\leq\omega}\omega$,

$$\iota_{\mathrm{B}}(y) := \begin{cases} \varnothing & \text{if } \forall i\ \exists j{>}i\ y(j) = \mathsf{B} \\ \iota_{\mathrm{W}}(y \to i) & \text{if } i \text{ is least such that } \forall j{\geq}i\ y(j) \neq \mathsf{B} \end{cases}$$

We define $\bar{\tau}$ as usual and we say that a Backtrack strategy τ is **winning** if $\bar{\tau} = f(x)$ for all $x \in A$.

Examples. Suppose Player II plays a sequence that contains infinitely many B's, then the interpretation will be $\varnothing$ and Player II can only win if Player I plays out of A. Suppose Player II plays the sequence

$$\langle a_0, a_1, \mathsf{B}, a_2, \mathsf{B}, a_3, a_4, \mathsf{P}, a_5, \dots\rangle$$

with two B's only, then the interpretation will be

$$\langle a_3, a_4, a_5, \dots\rangle.$$

By a theorem of Andretta [An06b, Theorem 20], the Backtrack game characterizes the $\mathcal{P}(\mathbf{\Pi}^0_1, \mathcal{F}(2,2))$ functions.

Theorem 4.1 (Andretta)**.** *Let $A \subseteq {}^\omega\omega$. A function $f : A \to {}^\omega\omega$ is $\mathcal{P}(\mathbf{\Pi}^0_1, \mathcal{F}(2,2)) \Leftrightarrow$ Player II has a winning strategy in the game $G_{\mathrm{B}}(f)$.*

Proof. $\Leftarrow$: Assume that τ is winning for Player II in the game $G_{\mathrm{B}}(f)$. Since, in the Baire space, every $\mathbf{\Sigma}^0_2$ set is the disjoint union of countably many $\mathbf{\Pi}^0_1$ sets (Lemma 1.1), it suffices to give a $\mathbf{\Sigma}^0_2$ partition. Let $A_n := \{x \in A :$ on input x, τ backtracks n times$\}$. Then it follows that $f{\restriction}A_n$ is continuous using Theorem 3.1. Namely, let τ' be the following Wadge strategy for Player II: "On a scratch tape, run τ until τ has backtracked n times. Then use the remaining output of τ as the output for τ'." It is clear that τ' is winning for Player II in the game $G_{\mathrm{W}}(f{\restriction}A_n)$, so $f{\restriction}A_n$ is continuous. Moreover, it is not difficult to see that A_n is $\mathbf{\Sigma}^0_2$ (in the relative topology of A). Let $\rho : {}^{<\omega}(\omega \cup \{\mathsf{P}, \mathsf{B}\}) \to \omega$ be defined by $\rho(s) :=$"the number of B's appearing in s". Then the formula

$$\exists i\ \forall j{>}i\ (\rho(\tau(x{\restriction}j)) = n)$$

witnesses that A_n is $\mathbf{\Sigma}^0_2$.

$\Rightarrow$: Let $\langle A_n : n \in \omega\rangle$ be given and let τ_n be a winning strategy for Player II in the game $G_{\mathrm{W}}(f{\restriction}A_n)$. The following strategy is easily seen to be winning for Player II in the game $G_{\mathrm{B}}(f)$: "Let $i := 0$. Run the Wadge strategy τ_i. If Player I plays out of A_i, then this is known after some finite period since the complement of A_i is open. Use the backtrack option and repeat the process with $i := i + 1$."

Q.E.D.

By a theorem of Jayne and Rogers, we conclude that the Backtrack game characterizes the $\mathcal{F}_{\mathrm{T}}(2,2)$ functions, meaning that a total function f is $\mathcal{F}(2,2)$ if and only if Player II has a winning strategy in the game $G_{\mathrm{B}}(f)$.

Theorem 4.2 (Jayne, Rogers)**.** *A function $f : {}^\omega\omega \to {}^\omega\omega$ is $\mathcal{F}(2,2) \Leftrightarrow$ there is a $\mathbf{\Pi}^0_1$ partition $\langle A_n : n \in \omega\rangle$ of ${}^\omega\omega$ such that $f{\restriction}A_n$ is continuous. In other words, $\mathcal{F}_{\mathrm{T}}(2,2) = \mathcal{P}_{\mathrm{T}}(\mathbf{\Pi}^0_1, \mathcal{F}(1,1))$.*

The original proof may be found in [Ja$_4$Ro$_1$82], and a more recent proof using embeddings may be found in [So$_1$98]. [So$_1$98] also provides a detailed analysis of the $\mathcal{F}(1,2)$ functions.

5 The Eraser game

In the Eraser game $G_{\mathrm{E}}(f)$, Player II plays elements of $\omega \cup \{\mathsf{E}\}$, with the token E interpreted to mean "erase." This option allows Player II to erase his most recent move in ω. In contrast with the backtrack option, it is possible for Player II to erase infinitely many times and still play an infinite sequence.

To define the interpretation function ι_{E}, we first define $\eta : {}^{<\omega}(\omega \cup \{\mathsf{E}\}) \to {}^{<\omega}\omega$ by recursion. Let $\eta(\varnothing) := \varnothing$ and let

$$\eta(s^\frown\langle z\rangle) := \begin{cases} \eta(s)^\frown\langle z\rangle & \text{if } z \in \omega, \\ \eta(s){\restriction}(\mathrm{lh}(\eta(s)) - 1) & \text{if } z = \mathsf{E} \text{ and } \mathrm{lh}(\eta(s)) > 0, \\ \varnothing & \text{otherwise.} \end{cases}$$

We then define $\iota_{\mathrm{E}} : {}^{\omega}(\omega \cup \{\mathsf{E}\}) \to {}^{\leq\omega}\omega$, $\iota_{\mathrm{E}}(y)(n) := m$ if $\exists i\ \forall j{>}i$, $\eta(y{\restriction}j)(n)$ is defined and equal to m. We define $\bar{\tau}$ as usual and say that an Eraser strategy τ is winning for Player II if $\bar{\tau}(x) = f(x)$ for all $x \in A$.

Examples. Suppose Player II plays the sequence

$$\langle a_0, a_1, a_2, \mathsf{E}, a_3, a_4, \dots\rangle$$

with one E only, then the interpretation will be

$$\langle a_0, a_1, a_3, a_4, \dots\rangle.$$

Suppose Player II plays

$$\langle a_0, a_1, a_2, \mathsf{E}, a_3, \mathsf{E}, a_4, \mathsf{E}, a_5, \mathsf{E}, \dots, a_i, \mathsf{E}, \dots\rangle$$

then the interpretation will be

$$\langle a_0, a_1\rangle$$

and Player II can only win the game if Player I plays out of A.

The Eraser game and its characterization (Theorem 5.1) are unpublished results due to Jacques Duparc.

Theorem 5.1 (Duparc)**.** Let $A \subseteq {}^{\omega}\omega$. A function $f : A \to {}^{\omega}\omega$ is $\mathcal{F}(1,2)$ $\Leftrightarrow$ Player II has a winning strategy in the game $G_{\mathrm{E}}(f)$.

The proof of Theorem 5.1 relies on the following topological fact about $\mathcal{F}(1,2)$ functions (see [Ke$_0$95, Theorem (24.10)]):

Theorem 5.2. A function $f : A \to {}^{\omega}\omega$ is $\mathcal{F}(1,2)$ $\Leftrightarrow$ f is the limit of a sequence of continuous functions $f_n : A \to {}^{\omega}\omega$.

Proof of Theorem 5.1. By Theorem 5.2, it suffices to show that f is the limit of a sequence of continuous functions f_n $\Leftrightarrow$ Player II has a winning strategy in the game $G_{\mathrm{E}}(f)$.

$\Rightarrow$: Let f_n be given and let τ_n be a winning strategy in the game $G_{\mathrm{W}}(f_n)$. Let τ be the following eraser strategy for Player II: "Let $x \in A$ be the play of Player I. Let $i := 0$. On a scratch tape, run τ_i until the first i elements of $\bar{\tau}_i(x)$ are determined. Then output these elements starting from the beginning of the tape, erasing only if necessary. Repeat the process with $i := i + 1$."

$\Leftarrow$: Let τ be winning for Player II is the game $G_{\mathrm{E}}(f)$. It suffices to give a sequence of Wadge strategies τ_n such that $\bar{\tau}_n(x)$ is infinite for all $x \in A$ and f is the limit of the $\bar{\tau}_n$. Let τ_n be the following: "Let $x \in A$ be the play of Player I. On a scratch tape, determine $\eta(\tau(x{\upharpoonright}n))$ (using the pass option until this is known). Then output $\eta(\tau(x{\upharpoonright}n))$, followed by random moves in ω."

Q.E.D. (Theorem 5.1)

6 The Multitape game

In the Multitape game $G_{\mathrm{M}}(f)$, Player II plays elements of $\omega \cup \{\uparrow, \downarrow\}$. We think of Player II as having countably many *rows* (or *tapes*), which he can select using $\uparrow$ and $\downarrow$. We associate a natural number to each row, with Player II starting at row 0 at the beginning of the game. At any point in the game, if Player II is on row n, he may move to row $n + 1$ using $\uparrow$ and row $n - 1$ using $\downarrow$. (If he is on row 0, the $\downarrow$ option has no effect.) So, each integer move occurs on some row. The interpretation is then the infinite sequence on the least row (if it exists) containing an infinite sequence. We refer to this row as the *output row.*

Formally, we define the interpretation function as follows. First, define $\rho : {}^{<\omega}(\omega \cup \{\uparrow, \downarrow\}) \to \omega$ as the "row" function on finite sequences of tokens. Let $\rho(\varnothing) := \varnothing$ and let

$$\rho(s^\frown\langle z\rangle) := \begin{cases} \rho(s) & \text{if } z \in \omega, \\ \rho(s)+1 & \text{if } z = \uparrow, \\ \rho(s)-1 & \text{if } z = \downarrow \text{ and } \rho(s) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

On infinite sequences of tokens, we define a partial function

$$\delta : {}^{\omega}(\omega \cup \{\uparrow, \downarrow\}) \to \omega$$

by letting $\delta(x)$ be the least $n \in \omega$ (if it exists) such that $\forall i\ \exists j{>}i$, $\rho(x{\upharpoonright}j) = n$ and $x(j-1) \in \omega$. In other words, the value of δ is the output row.

We next define $\zeta_n : {}^{<\omega}(\omega \cup \{\uparrow, \downarrow\}) \to {}^{<\omega}\omega$ by recursion. Let $\zeta_n(\varnothing) := \varnothing$ and let

$$\zeta_n(s^\frown\langle z\rangle) := \begin{cases} \zeta_n(s)^\frown\langle z\rangle & \text{if } z \in \omega \text{ and } \rho(s) = n, \\ \zeta_n(s) & \text{otherwise.} \end{cases}$$

In words, given a finite sequence of tokens, ζ_n is the output on the nth row. We may then define the interpretation function $\iota_{\mathrm{M}} : {}^\omega(\omega \cup \{\uparrow, \downarrow\}) \to {}^{\leq\omega}\omega$ by

$$\iota_{\mathrm{M}}(y) := \bigcup_{n\in\omega} \zeta_{\delta(y)}(y{\restriction}n) \text{ if } \delta(y) \text{ is defined.}$$

If $\delta(y)$ is undefined, we define $\iota_{\mathrm{M}}(y) := \varnothing$. We define $\bar{\tau}$ as usual and say that an Multitape strategy τ is winning for Player II if $\bar{\tau}(x) = f(x)$ for all $x \in A$.

Examples. Suppose Player II plays the sequence

$$\langle a_0, \uparrow, a_1, \uparrow, a_2, \uparrow, a_3, \uparrow, \ldots, a_i, \uparrow, \ldots\rangle,$$

then the interpretation will be $\varnothing$ since Player II has only played a single element on each row. Suppose Player II plays a sequence

$$\langle a_0, a_1, \uparrow, a_2, \uparrow, a_3, a_4, \downarrow, a_5, a_6, \ldots\rangle$$

such that the output row is row 1, then the interpretation will be

$$\langle a_2, a_5, a_6, \ldots\rangle.$$

Theorem 6.1 (Andretta, S.)**.** Let $A \subseteq {}^\omega\omega$. A function $f : A \to {}^\omega\omega$ is $\mathcal{P}(\mathbf{\Pi}^0_2, \mathcal{F}(1,1)) \Leftrightarrow$ Player II has a winning strategy in the game $G_{\mathrm{M}}(f)$.

Proof. $\Leftarrow$: Let τ be the winning multitape strategy for Player II. Since every $\mathbf{\Sigma}^0_3$ set in the Baire Space is the disjoint union of $\mathbf{\Pi}^0_2$ sets (Lemma 1.1), it suffices to give a $\mathbf{\Sigma}^0_3$ partition.

We define $A_n := \{x \in A : \delta(\hat{\tau}(x)) = n\}$. Note that $\delta(\hat{\tau}(x))$ is always defined if $x \in A$ since τ is winning. Thus $\langle A_n : n < \omega\rangle$ is indeed a partition of A. The following formula witnesses that A_n is $\mathbf{\Sigma}^0_3$ (in the relative topology):

$$\begin{aligned} &\exists i\, \forall j{>}i\, [\rho(\tau(x{\restriction}j)) < n \Rightarrow \tau(x{\restriction}j)(j{-}1) \in \{\uparrow, \downarrow\}] \wedge \\ &\qquad \forall i\, \exists j{>}i\, [\rho(\tau(x{\restriction}j)) = n \ \wedge\ \tau(x{\restriction}j)(j{-}1) \in \omega)]. \end{aligned}$$

In words, this formula says "there exists a round in the game after which Player II does not play an element of ω on a row less than n, and Player II plays infinitely many elements of ω on row n."

Furthermore, $f{\restriction}A_n$ is continuous. The following strategy is winning in $G_{\mathrm{W}}(f{\restriction}A_n)$: "Run τ on a scratch tape. Copy the output from row n, using the pass option when necessary."

$\Rightarrow$: Let A_n be given. Since A_n is $\mathbf{\Pi}^0_2$, there are $\mathbf{\Pi}^0_2$ formulas $\chi_n(x)$ (possibly with extra parameters) such that $x \in A_n \Leftrightarrow \chi_n(x)$. Since χ_n is $\mathbf{\Pi}^0_2$, we have that

$$\chi_n(x) \equiv \forall i\ \exists j\ \psi_n(x, i, j)$$

where ψ_n is a basic formula. (By a basic formula, we mean a formula in the language of arithmetic with bounded quantifiers only, cf. [Ka$_1$03, p. 152].)) Since ψ_n is a basic formula, for any i and j we can check whether $\psi_n(x, i, j)$ is true using only a finite initial segment of x. Let $x \in {}^{\omega}\omega$ be the infinite play of Player I. We are ready to define the multitape strategy τ for Player II. For each row n, there will be two counters, i_n and j_n, which are initialized to 0. At each stage of the game, Player II is considered to be *working* on a row n. We say that Player II works on a row n for one step if he does the following:

Given i_n and j_n, try to determine whether the formula $\psi_n(x, i_n, j_n)$ is true. Since only a finite initial segment of x is known, it may be impossible to do this, in which case do nothing. If the formula is true, run the Wadge strategy σ_n for one step on row n. Increment the i_n counter by 1, and reset the j_n counter to 0. If the formula is false, increment the j_n counter by 1.

In the obvious way, Player II can work on every row n for infinitely many steps. Since $\langle A_n \in \omega\rangle$ is a partition, Player I will play into exactly one A_n. This means that exactly one of the formulas $\chi_n(x)$ will be true, which means that Player II will only play infinitely often (namely, the Wadge strategy σ_n) on row n. Then $\bar{\tau}(x) = \bar{\sigma}_n(x) = f{\restriction}A_n(x) = f(x)$, so τ is winning.

Q.E.D.

7 The Multitape Eraser game

The Multitape Eraser game $G_{\mathrm{ME}}(f)$ is like the multitape game, except that Player II is given the additional option of erasing. So, Player II plays elements of $\omega \cup \{\uparrow, \downarrow, \mathsf{E}\}$. The output of Player II is the sequence on the least row (if it exists) on which Player II plays infinitely often in $\omega \cup \{\mathsf{E}\}$. Note that this sequence may not necessarily be infinite.

The definition of the interpretation function is similar to the case of the Multitape game. For convenience, we reuse some of the variables.

Define $\rho : {}^{<\omega}(\omega \cup \{\uparrow, \downarrow, \mathsf{E}\}) \to \omega$ as the "row" function on finite sequences of tokens. Let $\rho(\varnothing) := \varnothing$ and let

$$\rho(s^\frown\langle z\rangle) := \begin{cases} \rho(s) & \text{if } z \in \omega \cup \{\mathsf{E}\}, \\ \rho(s)+1 & \text{if } z = \uparrow, \\ \rho(s)-1 & \text{if } z = \downarrow \text{ and } \rho(s) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

On infinite sequences of tokens, define a partial function

$$\delta : {}^{\omega}(\omega \cup \{\uparrow, \downarrow, \mathsf{E}\}) \to \omega$$

by letting $\delta(x)$ be the least $n \in \omega$ (if it exists) such that $\forall i\ \exists j{>}i$, $\rho(x{\restriction}j) = n$ and $x(j-1) \in \omega \cup \{\mathsf{E}\}$. In other words, the value of δ is the output row.

We next define $\zeta_n : {}^{<\omega}(\omega \cup \{\uparrow, \downarrow, \mathsf{E}\}) \to {}^{<\omega}(\omega \cup \{\mathsf{E}\})$ by recursion. Let $\zeta_n(\varnothing) := \varnothing$ and let

$$\zeta_n(s^\frown\langle z\rangle) := \begin{cases} \zeta_n(s)^\frown\langle z\rangle & \text{if } z \in \omega \cup \{\mathsf{E}\} \text{ and } \rho(s) = n, \\ \zeta_n(s) & \text{otherwise.} \end{cases}$$

In words, given a finite sequence of tokens, ζ_n is the output (with the E's still present) on the nth row. We may then define the interpretation function $\iota_{\mathrm{ME}} : {}^{\omega}(\omega \cup \{\uparrow, \downarrow, \mathsf{E}\}) \to {}^{\leq\omega}\omega$ by

$$\iota_{\mathrm{ME}}(y) := \iota_{\mathrm{E}}\Big(\bigcup_{n\in\omega} \zeta_{\delta(y)}(y{\restriction}n)\Big) \text{ if } \delta(y) \text{ is defined.}$$

If $\delta(y)$ is undefined, then we define $\iota_{\mathrm{ME}}(y) := \varnothing$.

We define $\bar{\tau}$ as usual and say that a Multitape Eraser strategy τ is winning for Player II if $\bar{\tau}(x) = f(x)$ for all $x \in A$.

Example. Suppose Player II plays a sequence

$$\langle a_0, a_1, \uparrow, a_2, a_3, a_4, \mathsf{E}, a_5, \mathsf{E}, a_6, \mathsf{E}, \dots, a_i, \mathsf{E}, \dots\rangle$$

such that the output row is row 1, then the interpretation will be

$$\langle a_2, a_3\rangle.$$

Note that in this case, Player II can only win the game if Player I plays out of A.

Theorem 7.1. Let $A \subseteq {}^{\omega}\omega$. A function $f : A \to {}^{\omega}\omega$ is $\mathcal{P}(\mathbf{\Pi}^0_2, \mathcal{F}(1,2)) \Leftrightarrow$ Player II has a winning strategy in the game $G_{\mathrm{ME}}(f)$.

Proof. $\Leftarrow$: Let τ be the winning multitape strategy for Player II. We define $A_n := \{x \in A : \delta(\hat{\tau}(x)) = n\}$. As in the proof of Theorem 6.1, it follows that A_n is $\mathbf{\Sigma}^0_3$.

Furthermore, $f{\restriction}A_n$ is $\mathcal{F}(1,2)$. The following strategy is winning in $G_{\mathrm{E}}(f{\restriction}A_n)$: "Run τ on a scratch tape. Copy the output from row n, using the Eraser option when necessary."

$\Rightarrow$: As in Theorem 6.1, with "Wadge strategy" replaced by "Eraser strategy."

Q.E.D.

8 Future directions

We finish by noting several problems that are open (at least, as far as this author is aware). We state without proof the following:

Proposition 8.1. Let $m, n \geq 1$. Then $\mathcal{F}(n,m) \subseteq \mathcal{F}(n+1, m+1)$. Therefore, $\mathcal{F}(n,m) \subseteq \mathcal{F}(n+k, m+k)$ for any $k \geq 0$.

The following fact is also easy to show:

Proposition 8.2. Let $m, n \geq 2$. Then $\mathcal{P}(\mathbf{\Pi}^0_{m-1}, \mathcal{F}(1, m-n+1)) \subseteq \mathcal{F}(n,m)$.

Proof. Let $f : A \to {}^{\omega}\omega$ in $\mathcal{P}(\mathbf{\Pi}^0_{m-1}, \mathcal{F}(1, m-n+1))$ and let $\langle A_i : i < \omega\rangle$ be the partition. Let $Y \in \mathbf{\Sigma}^0_n$ and $Y_j \in \mathbf{\Pi}^0_{n-1}$ such that $Y = \bigcup_j Y_j$. It follows that

$$\begin{aligned} f^{-1}[Y] &= \bigcup_i (f{\restriction}A_i)^{-1}[Y] \\ &= \bigcup_i \bigcup_j (f{\restriction}A_i)^{-1}[Y_j] \\ &= \bigcup_i \bigcup_j A \cap X_{ij}, \text{ where } X_{ij} \in \mathbf{\Pi}^0_{m-1} \\ &= A \cap X, \text{ where } X \in \mathbf{\Sigma}^0_m. \end{aligned}$$

For the second to last equality, note that $f{\restriction}A_i \in \mathcal{F}(n-1, m-1)$ by Proposition 8.1 (take $k = n-2$). Q.E.D.

Problem 8.3. For which m and n is

$$\mathcal{F}_{\mathrm{T}}(n,m) = \mathcal{P}_{\mathrm{T}}(\mathbf{\Pi}^0_{m-1}, \mathcal{F}(1, m-n+1))?$$

In particular, the statement for $n = m = 2$ is Theorem 4.2. If the statement were to hold for $n = m = 3$, then the Multitape game would characterize the total $\mathcal{F}(3,3)$ functions. Similarly, if the statement were to hold for $n = 2$ and $m = 3$, then the Multitape Eraser game would characterize the total $\mathcal{F}(2,3)$ functions.

Problem 8.4. Is the inclusion in Proposition 8.1 always proper?

Problem 8.5. For $n \geq 2$ and $m \geq 1$, is $\mathcal{F}(n,m)$ properly contained in $\mathcal{F}(n-1,m)$?

References

[An06b] A. Andretta. More on Wadge determinacy. *Annals of Pure and Applied Logic* 144(1–3):2–32, 2006.

[Ja$_4$Ro$_1$82] J.E. Jayne & C.A. Rogers. First level Borel functions and isomorphisms. *Journal de Mathématiques Pures et Appliquées* 61(2):177–205, 1982.

[Ka$_1$03] A. Kanamori. *The Higher Infinite. Large Cardinals in Set Theory from Their Beginnings.* Monographs in Mathematics, Springer, 2003.

[Ke$_0$95] A.S. Kechris. *Classical Descriptive Set Theory*, *Graduate Texts in Mathematics* 156. Springer, 1995.

[Ke$_0$Mo$_1$78] A.S. Kechris & Y.N. Moschovakis, eds. *Cabal Seminar 76–77 (Proceedings of the Caltech-UCLA Logic Seminar, 1976–77)*, *Lecture Notes in Mathematics* 689. Springer, 1978.

[Mo$_1$80] Y.N. Moschovakis. *Descriptive Set Theory.* North-Holland, 1980.

[So$_1$98] S. Solecki. Decomposing Borel sets and functions and the structure of Baire class 1 functions. *Journal of the American Mathematical Society* 11(3):521–550, 1998.

[VW78] R. Van Wesep. Wadge degrees and descriptive set theory. In [Ke$_0$Mo$_1$78, pp. 151–170].

[Wa$_0$83] W.W. Wadge. *Reducibility and determinateness on the Baire space.* Ph.D. thesis, University of California, Berkeley, 1983.

The Complexity of Scotland Yard[*]

Merlijn Sevenster

Philips Research
Prof. Holstlaan 4
5656 AA Eindhoven, The Netherlands
merlijn.sevenster@philips.com

Abstract

Games with imperfect information have escaped the interest of researchers in combinatorial game theory. This paper discusses a computational case study of the board game of Scotland Yard. Interestingly, Scotland Yard is a genuine “playgame” with imperfect information. We show by means of a powerset argument, that Scotland Yard can also be considered a game of perfect information, that is isomorphic to the imperfect information variant. Using the powerset analysis, we show that Scotland Yard has **PSPACE**-complete complexity be it with or without imperfect information. In fact, imperfect information may even simplify matters: if the cops are supposed to be consequently ignorant of Mr. X’s whereabouts throughout the game the complexity is **NP**-complete.

1 Introduction

1.1 Background

The discipline of *combinatorial game theory* (CGT) deals almost exclusively with zero-sum games with perfect information. Although the existence of games with imperfect information is acknowledged in one of CGT’s seminal publications [$\text{Be}_1\text{Co}_3\text{Gu}_2$82, pp. 16–7], only a marginal amount of literature appeared on games with imperfect information. Yet, the number of publications on games with perfect information is abundant and offers a robust picture of the computational behavior of games: One-person games

[*] I wish to thank the organizers of the seventh August de Morgan workshop for inviting me to present my research. I gratefully acknowledge Peter van Emde Boas for carefully proofreading an earlier version, and Ulle Endriss for his remarks that lead to Section 7. I’m greatly indebted to the anonymous referees whose comments helped to improve the presentation of this paper. I thank Berend ter Borg for repairing my English in parts of this paper.
Finally, let me acknowledge Victor de Boer, Berend ter Borg, and Sicco Kuijper with whom I played many games of Scotland Yard on Sunday afternoons, already more than ten years ago.

or *puzzles* are usually solvable in **NP** and many of them turn out to be complete for this class.[1] Famous examples include the games of Minesweeper [$\mathrm{Ka}_5$00] and Clickomania [Bi_0+02]. Alternation increases complexity considerably: many two-player games have **PSPACE**-hard complexity, such as Go [$\mathrm{Li}_1\mathrm{Si}_2$80] and the semantic evaluation game of *quantified boolean formulae* [$\mathrm{St}_6\mathrm{Me}_2$73, $\mathrm{Sc}_0$78]. Some even have **EXPTIME**-complete complexity. Typical examples in this respect are the games of Chess [$\mathrm{Fr}_0\mathrm{Li}_1$81] and Checkers [$\mathrm{Fr}_0$+78]. By and large, games with **EXPTIME**-complete complexity have a *loopy* nature, that is, the same configuration may occur over and over again. In real-life, loopy games may not be that much fun to play, as they allow for annoyingly long runs in which neither player makes any progress. Loopy runs are banned from Chess by postulating that, roughly speaking, no configuration of the game may occur more than three times.

Amusingly, putting an upper-bound on the duration of the game not only avoids loopy sequences of play, but also has considerable computational impact. Papadimitriou [$\mathrm{Pa}_4$94, pp. 460–2] argues that every game that meets the following requirements is solvable in **PSPACE**:

- the length of any legal sequence of moves is bounded by a polynomial in the size of the input; and
- given a "board position" of the game there is a polynomial-space algorithm which constructs all possible subsequent actions and board positions; or, if there aren't any, decides whether the board position is a win for either player.

Note that Papadimitriou does not even mention the fact that this result concerns games of perfect information. The result stands due to the fact that the backwards induction algorithm (see also Section 2) can be run on the game's game tree in **PSPACE**, given that it meets the above requirements.

As for games of imperfect information, some studies have been performed and they invariable report an increase of complexity. Convincing results in this vein are given in [$\mathrm{Ko}_3\mathrm{Me}_0$92], where the authors show that it is possible to decide whether either player has a winning strategy in a finite, two-player game of perfect information using a polynomial time in the size of the game tree. On a positive note, they show that there is a **P**-algorithm that solves the same problem for games of imperfect information with perfect recall. However, if one of the players suffers from imperfect recall the problem of deciding whether this player has a winning strategy is **NP**-hard in the size of the game tree.

In [$\mathrm{Re}_1$84, $\mathrm{Pe}_3\mathrm{AzRe}_1$01] the authors regard computation trees as game trees. This view on computation trees is adopted from [$\mathrm{Ch}_0\mathrm{Ko}_6\mathrm{St}_6$81], in

[1] To solve a game is to determine for an instance of the game whether a designated player has a winning strategy.

which so-called *alternating Turing machines* are considered which have existential and universal states. The aspect of alternation is reflected in the computation tree by regarding it as a game tree of a two-player game. The nodes corresponding to existential (universal) states belong to the existential (universal) player. From this viewpoint, non-deterministic Turing machines have no universal states and thus give rise to one-player game trees. In [$\mathrm{Re}_1$84, $\mathrm{Pe}_3\mathrm{AzRe}_1$01] this idea is extended to games of imperfect information. The authors define *private alternating Turing machines*, which give rise to computation trees that may be regarded as two-player game trees in which the existential player suffers from imperfect information, among other devices. It is shown that the space complexity of $f(n)$ of these machines is characterized in terms of the complexity of alternating Turing machines with space bound exponential in $f(n)$. Moreover, it is shown that private alternating Turing machines with three players—with two of the players teaming up—can recognize undecidable problems in constant space.

Dramatic as these results may be, being general studies they cannot tell us what the computational impact of the imperfect information found in actual games is. That is, games developed to be played rather than to be analyzed.[2] It may well turn out that the imperfect information in these games has little computational impact and that the games themselves match the robust intuitions we have about the computational nature of perfect information games. As we pointed out before, there is but a small number of results concerning games with imperfect information, let alone computational studies of parlor games. For this reason, we shall consider the game of Scotland Yard which gamers have enjoyed since 1983.[3] Readers familiar with Scotland Yard will acknowledge that it is the imperfect information that makes the game an enjoyable waste of time and enthusiastic accounts of players' experiences with Scotland Yard are readily found on the Internet.

1.2 Game rules

We shall now give an outline of the rules of Scotland Yard. Note however that this description serves mainly to stress the kinship between the formalization used in this paper and the actual game. A complete set of game rules of the formalization is supplied in the next section and suffices to understand the formal details.

Scotland Yard is played on a game board which contains approximately 200 numbered intersections of colored lines denoting available means of

[2] Fraenkel [$\mathrm{Fr}_0$02, p. 476] makes the distinction between "*PlayGames*" and "*MathGames*". The former being the games that "are challenging to the point that people will purchase them and play them", whereas the latter games "are challenging to a mathematician [. . .] to play with or ponder about."

[3] Scotland Yard is produced by *Ravensburger/Milton Bradley* and was prestigiously declared *Spiel des Jahres* in 1983.

FIGURE 1. The box of Scotland Yard and its items, amongst which the game board, Mr. X's move board, and the players' pawns. This picture is reproduced with permission of *Ravensburger*.

transportation: yellow for taxis, green for buses, and pink for the Underground. A game is played by two to six people, one of them being *Mr. X*, the others teaming up and thusly forming Scotland Yard. They have a shared goal: capturing Mr. X. Initially, every player gets assigned a pawn and an intersection on the game board on which his or her pawn is positioned. Before the game starts every player gets a fixed number of tickets for every means of transportation. Mr. X and the cops move alternatingly, Mr. X going first.

During every stage of the game, each player—Mr. X or his adversaries—takes an intersection in mind connected to his or her current intersection, subject to him or her owning at least one ticket of the appropriate kind. For instance, if a player would want to use the metro from Buckingham Palace, she would have to hand in her metro ticket. If either player is out of tickets for a certain means of transportation, he cannot travel along the related lines. Every player's set of tickets is known to all players at every stage of the game. No player is allowed to stand still for one or more round. For cops it is not allowed to share the same node; in case a cop gets stuck, i.e., she cannot move to any node from her current position, she is out of the game.[4]

[4] The latter two conditions are ignored in this paper's abstraction. We suspect they would not affect the computation results, though.

Scotland Yard is sold as a 2-to-6 player game. But nothing essential gets lost when one player controls several or all cops pawns, since communication between the cops is allowed at all times.

If a cop has made up her mind to move to an intersection, this is indicated by her moving the pawn under her control to the intersection involved. However, when Mr. X has made up his mind he secretly writes the intersection's number in the designated entry of the *move board* and covers it with the ticket he has used. The cops know what means of transportation Mr. X has been using, but do not know his position. After round 3, 8, 13, 18, and 24, however, Mr. X is forced to show his whereabouts by putting his pawn on his current hideout.

The game lasts for 24 rounds during which Mr. X and the cops make their moves. If at any stage of the game any of the cops is at the same intersection as Mr. X the cops win.[5] Since Mr. X is invisible most of the time, he has to actually reveal himself once caught. It is rather pointless for Mr. X to not announce his loss, as the cops can reconstruct his path using the game board when the game has ended. If Mr. X remains uncaught until after the last round, he wins the game. Cops who have a suspicious nature may want to check whether Mr. X's secret moves were consistent with the lines on the game board when the game is over. To this end, they would match the numbers on the move board with the returned tickets. If it turns out that Mr. X has cheated at any point or has actually been caught, he loses no matter what the outcome of the game.

In view of these game descriptions, the generalization of the Scotland Yard game in Definition 1.1 may strike the reader as a natural abstraction. The reader will observe that the number of means of transportation is reduced to one and that the game board is modelled by a directed graph. However, all results in this paper can be taken to hold for instances where several means of transportation and undirected graphs are involved.

Definition 1.1. Let $G = \langle V, E\rangle$ be a finite, connected, directed graph with an out-degree of one or higher. That is, for every $v \in V$, there is a $v' \in V$, such that $E(v, v')$. Let $u, v_1, \ldots, v_n \in V$. Let $f : \{1, \ldots, k\} \to \{\mathsf{show}, \mathsf{hide}\}$ be the *information function*, for some integer $2 < k < |V|$. Then, let $\langle G, \langle u, v_1, \ldots, v_n\rangle, f\rangle$ be a *(Scotland Yard) instance*. Most of the time it will be convenient to abbreviate a string of vertices $v_1, \ldots, v_n$ by $\vec{v}$. Conversely, $\vec{v}(i)$ shall denote the ith element in $\vec{v}$. By $\{\vec{v}\}$ we refer to the set of all vertices in $\vec{v}$.

For $U \subseteq V$ write $E(U)$ to denote $\{u' \in V \mid E(u, u')$, for some $u \in U\}$. If $\vec{v}, \vec{v}' \in V^n$, then write $E(\vec{v}, \vec{v}')$ to denote that for every $1 \le i \le n$, $E(\vec{v}(i), \vec{v}'(i))$.

[5] Note that this is the description of the actual Scotland Yard game. My formalization—to be provided—has slightly different winning conditions.

The information function f controls the imperfect information throughout the game. If round i has property $f(i) = \mathsf{hide}$, Mr. X hides himself. As will be seen the information function gives an intuitive meaning to "adding" and "removing" imperfect information from a Scotland Yard game. For instance, if one restricts oneself to information functions with range $\{\mathsf{show}\}$, Mr. X shows his whereabouts after every move and one is effectively considering a game of perfect information. Under the latter restriction, one has arrived at so-called *graph games*, also called *Pursuit* or *Cops and robbers.* For an exposition of the literature on graph games, consult [Go$_1$Re$_2$95].

1.3 Aims and structure

The aims of this paper are twofold. First to pinpoint the computational complexity of a real game of imperfect information. Secondly, we go through a reasonable amount of effort to spell out the relation between the game of Scotland Yard and a game of perfect information that is highly similar to the former. More precisely, we show that the games' game trees are isomorphic, and that a winning strategy in the one game constitutes a winning strategy in the other and vice versa. These similarity results may convince the reader that in some cases the wall between perfect and imperfect information is not as impenetrable as one might induce from the scarce literature on the complexity of imperfect information games.

As we pointed out before, the definition of Scotland Yard instance abstracts away from features of the game of Scotland Yard that are inessential to this paper's aims. In fact, all that a Scotland Yard instance holds is a graph, a set of vertices on which the pawns are initially positioned, the duration of the game, and a means to control the imperfect information. In my view, the level of abstraction employed in Definition 1.1 justifies one's conceiving Scotland Yard instances as graph games. For this reason, I think that my analyses are not solely relevant to the specific game of Scotland Yard, but to the theory of graph games in general. Nevertheless, I shall continue to refer to the games under consideration using the colloquial Scotland Yard terminology.

Section 2 reviews the basic notions from game theory and complexity theory and fixes notation. In Section 3, we shall define the extensive game form of the Scotland Yard game to which a Scotland Yard instance gives rise. In Section 4, we define a perfect information variant of Scotland Yard. In this perfect information game, Mr. X picks up sets of vertices, but he does so in public. In Section 5, we show that the games that have been introduced admit for a bijection between the imperfect information game's information partitions and the histories in the perfect information Scotland Yard game. Furthermore, we show that both games, under the bijection analysis, are equivalent, i.e., the cops have a winning strategy in the imperfect information game iff they have one in the perfect information game. In Section 6, the computational results are presented. In accordance with many polynomially bounded two-player games, Scotland Yard is complete for **PSPACE**, despite its imperfect information. That is, the computational complexity of Scotland Yard does not change when one only considers information functions with range $\{\mathsf{show}\}$.

In fact, if one would add more imperfect information to the extent that the information flow function has range $\{\mathsf{hide}\}$, the resulting decision problem is easier: **NP**-complete. This is shown in Section 7. Finally, Section 8 concludes the paper.

2 Preliminaries

In this section, we introduce some basics from game theory and complexity to make this paper reasonably self contained.

2.1 Game theory

Key notion in this paper is the notion of *extensive game with imperfect information.* One may consider this an extension of the notion of *extensive game with perfect information*, due to [vNMo$_0$44].

A win-loss extensive game with perfect information G is a four-tuple $\langle N, H, P, U\rangle$, where

- N is the set of *players.* Referring to the number of players, G is called an $\|N\|$*-player game.*

- H is a set closed under prefixes and denotes the set of *histories.* A history shall be regarded as an ordered list $\langle a_1, \ldots, a_n\rangle$ of *actions.* If $h = \langle a_1, \ldots, a_n\rangle$, then $\langle h, a'\rangle$ denotes $\langle a_1, \ldots, a_n, a'\rangle$ and is called an *immediate successor* of h. $A(h)$ denotes all actions that extend h. Let Z be the subset of H whose histories cannot be extended. Z is called the *set of terminal histories.*

- P is the *player function*, that assigns to every non-terminal history h a player $P(h)$. Formally, P is a function of type $(H-Z) \to N$. We say that a history h *belongs to* $P(h)$.

- U is the *utility function*, assigning to every terminal history one winning player.

Let G be a win-loss extensive game with perfect information as above. Then, a function S is called a *strategy* for player $i \in N$ in G, if it maps every history h belonging to i, to an action in $A(h)$. Let S be a strategy for player i in G. Then, call a history $h = \langle a_1, \ldots, a_n\rangle$ in *accordance* with S, if for every proper initial prefix $h' = \langle a_1, \ldots, a_k\rangle$ of h, where $k < n$, such that $P(h') = i$, it is the case that $\langle a_1, \ldots, a_k, S(h')\rangle$ is also an initial prefix of h. Furthermore, if for every history h that is in accordance with S it is the case that $U_i(h) = i$, then S is called a *winning strategy* for i in G.

Let G be a win-loss extensive game with perfect information, that is also two-player. The *backwards induction algorithm*, due to [Ze13], decides whether player i has a winning strategy. The algorithm takes G as input and goes about as follows:

- Label all terminal histories h in G with $U(h)$.
- Until the initial history ε has been labelled, consider every unlabelled, non-terminal history h in G and do as follows:
  - If $P(h) = i$ and there is an immediate successor history h' of h labelled i, then label h with i.
  - If $P(h) \neq i$ and every immediate successor history h' of h is labelled $j \in (N - \{i\})$, then label h with lose.

An easy inductive argument shows that the backwards induction algorithm labels the empty history with i iff player i has a winning plan of action in G. Papadimitriou [Pa$_4$94] showed that the backwards induction algorithm can be implemented in such a way that it consumes polynomial space in the size of the input, given that the game meets the aforementioned conditions.

Extensive games with imperfect information extend extensive games with perfect information in that they are five-tuples $\langle N, H, P, \langle \mathcal{I}_i\rangle_{i\in N}, U\rangle$, carrying *information sets* $\mathcal{I}_i$ for every player $i \in N$. The other notions are similar to the ones defined for extensive games with perfect information. An information set $\mathcal{I}_i = \{I_1, \ldots, I_n\}$ is a partition of the set of histories belonging to player i, that meets the *action consistency requirement*: if h and h' sit in $I \in \mathcal{I}_i$, then $A(h) = A(h')$. Every partition in an information set is called an *information partition*. If $I \in \mathcal{I}_i$, player i is said to *own* I and $\mathcal{I}_i$, or they are said to *belong* to i. Intuitively, an extensive game with imperfect information models the situation in which player i knows that some history $h \in I \in \mathcal{I}_i$ has happened, but she is unable to tell h apart from the other histories in I. The requirement that all histories in an information partition can be extended by the same actions—the action consistency requirement—captures the idea that otherwise the player owning the information partition could deduce information about the actual history from the actions available. If $I \in \mathcal{I}_i$, write $A(I)$ to denote $A(h)$, for an arbitrary $h \in I$.

Let $G = \langle N, H, P, \langle \mathcal{I}_i\rangle_{i\in N}, U\rangle$ be an extensive game with imperfect information. Then, a function S is called a *strategy* for player i in G, if it maps every information partition $I \in \mathcal{I}_i$ belonging to i onto an action in $A(I)$.

In the context of win-loss games, a strategy S for player i is called *winning in* G if every terminal history h that is in accordance with S yields $U(h) = i$.

2.2 Complexity theory

Complexity theory measures the complexity of a decision problem in terms of the requires resources, such as time and space. We refer the reader for an excellent introduction to this exciting field to [Pa$_4$94].

In this paper, the complexity classes **NP** and **PSPACE** play a central role. The notion of *polynomial-time computable many-one reduction* is simply referred to by *reduction.* Recall that if there is a decision problem to which every decision problem in a complexity class, say **NP**, is reducible, then this decision problem is **NP**-*hard.* If moreover this decision problem itself is solvable in **NP**, then we call it **NP**-complete—and likewise for **PSPACE**.

A convenient family of decision problems is related to propositional logic. In this paper, we use two logical decision problems: 3-SAT and QBF. The former is **NP**-complete, whereas the latter is **PSPACE**-complete.

To introduce the problems properly, let me introduce some folklore terminology from propositional logic. A *literal* is a propositional variable or a negated propositional variable. A *clause* is a disjunction of literals. A boolean formula is in *conjunctive normal form* (CNF), if it is a conjunction of clauses. A boolean formula is said to be in *3-CNF*, if it is in CNF and all of its clauses contain exactly three literals. The decision problem 3-SAT consists of every satisfiable 3-CNF formula φ, that is, every φ for which there exists a truth assignment to its variables that makes φ true.

The decision problem QBF has *quantified boolean formulae* as instances, i.e., objects of the form $\forall x_1 \exists y_1 \ldots \forall x_n \exists y_n \ \varphi$, in which φ is a boolean formula in 3-CNF over the variables $x_1, y_1, \ldots, x_n, y_n$. The problem QBF asks whether for every truth value for x_1, there is a truth value for y_1, and for every truth value for x_2, …, such that the boolean formula φ is satisfied by the resulting truth assignment.

3 Scotland Yard formalized

In this section, we define the extensive games with imperfect information that are constituted by Scotland Yard instances.

Let SY $= \langle G, \langle u_*, \vec{v}_* \rangle, f \rangle$ be a Scotland Yard instance as in Definition 1.1. Before we define the extensive game form of the Scotland Yard game to which SY gives rise, let me formulate the game rules of the game under consideration in terms of SY.

The digraph G is the board on which the actual playing finds place. In the initial situation of the game $n + 1$ pawns, named $\forall, \exists_1, \ldots, \exists_n$, are positioned on the respective vertices $u_*, \vec{v}_*(1), \ldots, \vec{v}_*(n)$ on the digraph. The game is played by the two players $\exists$ and $\forall$ over k rounds, and with every round $1 \leq i \leq k$ in the game the property $f(i) \in \{\mathsf{show}, \mathsf{hide}\}$ is associated. Note that we converted the n-player game of Scotland Yard, where $2 \leq n \leq 6$ into a two-player game in which one player controls all pawns $\exists_1, \ldots, \exists_n$. Furthermore, for reasons of succinctness we use the symbol $\forall$ to refer to Mr. X (male) and $\exists$ to refer to the player controlling Scotland Yard (female). Somewhat sloppily, sometimes we shall not make a strict distinction between a player and (one of) his or her pawns.

The initial positions of the players' pawns are known at the outset of the game. Fix $i = 1$, $u = u_*$, and $\vec{v} = \vec{v}_*$; round i of Scotland Yard goes as follows:

1. If for some $1 \leq j \leq n$, the pawns $\forall$ and $\exists_j$ share the same vertex, i.e., $u = \vec{v}(j)$, $\forall$ is said to be *captured* (by $\exists_j$). If $\forall$ is captured the game stops and $\exists$ wins. If $\forall$ is not captured and $i > k$ the game also stops but $\exists$ loses.

2. $\forall$ chooses a vertex u', such that $E(u, u')$. If $f(i) = \mathsf{show}$, $\forall$ physically puts his pawn on u'. If $f(i) = \mathsf{hide}$, he secretly writes u' on his move board making sure that it cannot be seen by his opponent. Set $u = u'$.

3. Player $\exists$ chooses a vector $\vec{v}' \in V^n$, such that $E(\vec{v}, \vec{v}')$, and for every $1 \leq j \leq n$, moves pawn $\exists_j$ to $\vec{v}'(j)$. Set $\vec{v} = \vec{v}'$.

4. Set $i = i + 1$.

Note that these game rules do not consider the possibility of either player getting stuck, as in not being able to move a pawn under his or her control along an edge. This goes without loss of generality, as the digraphs at stake are supposed to have out-degree ≥ 1.

Furthermore, it should be borne in mind, that for $\forall$ it is not a guaranteed loss to move to a vertex occupied by one of $\exists$'s pawns. The game only terminates after $\exists$ has moved *and* one of her pawns captures $\forall$, unlike the game rules for the board game of Scotland Yard.

Observe that $\exists$ loses only after k rounds of the game have been played. If it so happens that the game terminates after the jth round, for $j < k$, then $\exists$ has won.

Scotland Yard is modelled as an extensive game with imperfect information in Definition 3.1. The upcoming definition and Definition 4.1 are notationally akin to the definitions in Section 2.

Definition 3.1. Let $\textsc{sy} = \langle G, \langle u_*, \vec{v}_* \rangle, f \rangle$ be a Scotland Yard instance. Then, let the *extensive Scotland Yard game constituted by* $\textsc{sy}$ be defined as the tuple $\mathrm{SY}(\textsc{sy}) = \langle N, H, P, \sim, U \rangle$, where

- $N = \{\exists, \forall\}$.
- In order to define H, let ℓ add to every finite tuple $\langle a_1, \ldots, a_n \rangle$ the integer $(\lceil n/2 \rceil) - 1$. Then, H is the smallest set containing $\langle u_* \rangle$, $\langle u_*, \vec{v}_* \rangle$ and is closed under actions taken by $\forall$ and $\exists$:
  - If $\langle h, u, \vec{v} \rangle \in H$, $\ell(\langle h, u, \vec{v} \rangle) < k$, $u \notin \{\vec{v}\}$, and $E(u, u')$, then $\langle h, u, \vec{v}, u' \rangle \in H$.
  - If $\langle h, u, \vec{v}, u' \rangle \in H$ and $E(\vec{v}, \vec{v}')$, then $\langle h, u, \vec{v}, u', \vec{v}' \rangle \in H$.

  For $h \in H$, $\ell(h)$ denotes the number of rounds of h. For instance, $\ell(\langle u_*, \vec{v}_* \rangle) = 0$ and $\ell(\langle u_*, \vec{v}_*, u_1, \vec{v}_1, u_2 \rangle) = 2$. Somewhat unlike custom usage in game theory, the length $\ell(h)$ of history h does not coincide with the number of plies in the game. This notation is chosen to reflect the game rule saying that a history only terminates after $\exists$ has moved—that is, after every completed round.

  Let $\prec$ be the *immediate successor* relation on H. That is, the smallest relation closed under the following conditions:
  - If $h, \langle h, u \rangle \in H$, then $h \prec \langle h, u \rangle$.
  - If $\langle h, u \rangle, \langle h, u, \vec{v} \rangle \in H$, then $\langle h, u \rangle \prec \langle h, u, \vec{v} \rangle$.
- Due to the notational convention, the value of the player function P is easily determined from the history's form, in the sense that $P(\langle h, u \rangle) = \exists$ and $P(\langle h, u, \vec{v} \rangle) = \forall$, no matter h, u, and $\vec{v}$.
- $\sim$ is the *indistinguishability relation* that formalizes the imperfect information in the game. It is defined such that for any pair of histories $h, h' \in H$ of equal length, where

$$h = \langle u_*, \vec{v}_*, u_1, \vec{v}_1, \ldots, u_i \rangle \quad \text{and} \quad h' = \langle u_*, \vec{v}_*, u'_1, \vec{v}'_1, \ldots, u'_i \rangle \tag{1.1}$$

  it is the case that $h \sim h'$, if

  (a) $\vec{v}_j = \vec{v}'_j$, for every $1 \le j \le i - 1$; and

  (b) $u_j = u'_j$, for every $1 \le j \le i$ such that $f(j) = \mathsf{show}$.

  The previous condition, considering histories as in (1.1), defines $\sim$ only as a relation between histories h belonging to $\exists$. This reflects the fact that it is $\exists$ who experiences the imperfect information. Somewhat unusual, we extend $\exists$'s indistinguishability relation $\sim$ to histories in which $\forall$ has to move. The reader is urged to take this extension as a technicality. We put as follows: for any pair of histories $h, h' \in H$, where

$$h = \langle u_*, \vec{v}_*, u_1, \vec{v}_1, \ldots, u_i, \vec{v}_i \rangle \quad \text{and} \quad h' = \langle u_*, \vec{v}_*, u'_1, \vec{v}'_1, \ldots, u'_i, \vec{v}'_i \rangle \tag{1.2}$$

it is the case that $h \sim h'$, if

(a) $\vec{v}_j = \vec{v}'_j$, for every $1 \leq j \leq i$; and

(b) $u_j = u'_j$, for every $1 \leq j \leq i$ such that $f(j) = \mathsf{show}$.

- $U : Z \rightarrow \{\mathsf{win}, \mathsf{lose}\}$ is the function that decides whether a terminal history $\langle h, u, \vec{v} \rangle$ is won or lost for $\exists$. Formally,

$$U(\langle h, u, \vec{v} \rangle) = \begin{cases} \mathsf{win} & \text{if } u \in \{\vec{v}\} \\ \mathsf{lose} & \text{if } u \notin \{\vec{v}\}. \end{cases}$$

  Note the discrepancy with the definition of utility functions in Section 2. As this papers is only concerned with $\exists$'s having a winning strategy, we found the current utility function more convenient.

Since $\sim$ is reflexive, symmetric, and transitive it defines an equivalence relation on H. We write $\mathcal{H} \subseteq \wp(H)$ for the set of equivalence classes, or *information cells*, in which H is partitioned by $\sim$. That is, $\mathcal{H} = \{C_1, \ldots, C_m\}$, where $C_1 \cup \ldots \cup C_m = H$ and for every $1 \leq i \leq m$, if $h \in C_i$ and $h \sim h'$, then $h' \in C_i$. A standard inductive argument suffices to see that for every $C_i \in \mathcal{H}$ and pair of histories $h, h' \in C_i$, the length of h and h' coincides and $P(h) = P(h')$.

We lift the relation $\prec$ to $\mathcal{H}$, using the same symbol: For any pair $C, C' \in \mathcal{H}$, we write $C \prec C'$ if there exists histories $h \in C$ and $h' \in C'$ such that $h \prec h'$. It is easy to see that if h, h' are histories in a cell $C \in \mathcal{H}$, then $P(h) = P(h')$. Thus, the player function is meaningfully lifted as follows: if $C \in \mathcal{H}$ and h is a history in C, then $P(C) = P(h)$. Call a cell $C \in \mathcal{H}$ *terminal* if all its histories are terminal.

Since we study an extension of $\sim$, the set $\mathcal{H}$ partitions all histories in H. As we pointed out in the definition of $\sim$, if histories h and h' stand in the $\sim$ relation and belong to $\forall$, this should not be taken to reflect any conceptual consideration about $\exists$'s experiences. Yet, if h and h' belong to $\exists$, to write $h \sim h'$ reflects genuine indistinguishability for player $\exists$ between the two histories h and h'. Consider the set $\mathcal{H}_\exists = \{C \in \mathcal{H} \mid P(C) = \exists\}$, that partitions the set of histories that belong to $\exists$. We claim that $\mathcal{H}_\exists$ is an *information set* in the sense of Section 2, that is, it meets the action consistency requirement. A proof to this effect is straightforward.

The notion of winning strategy readily applies to the games SY(SY), despite the fact that $\exists$'s indistinguishability relation is modelled by $\sim$ and not by the customary $(\mathcal{I}_i)_{i \in N}$. We claim without proof that the "extended usage" of $\sim$ does not affect $\exists$'s having a winning strategy. To formulate this claim more precisely, let $G(\textsc{sy})$ model the Scotland Yard game constituted by $\textsc{sy}$ using the customary information partitions. Then, $\exists$ has a winning strategy in $G(\textsc{sy})$ iff she has one in $\mathrm{SY}(\textsc{sy})$.

For future reference, we lay down the following proposition.

Proposition 3.2. Let $\mathrm{SY}(\textsc{sy}) = \langle N, H, P, \sim, U\rangle$ be the Scotland Yard game constituted by $\textsc{sy}$. Then, the following statements hold:

(1) If $\langle h_1, u_1\rangle \sim \langle h_2, u_2\rangle$ and $f(\ell(\langle h_1, u_1\rangle)) = \mathsf{hide}$, then $h_1 \sim h_2$.

(2) If $\langle h_1, u_1\rangle \sim \langle h_2, u_2\rangle$ and $f(\ell(\langle h_1, u_1\rangle)) = \mathsf{show}$, then $h_1 \sim h_2$ and $u_1 = u_2$.

(3) If $\langle h_1, u_1, \vec{v}_1\rangle \sim \langle h_2, u_2, \vec{v}_2\rangle$, then $\langle h_1, u_1\rangle \sim \langle h_2, u_2\rangle$ and $\vec{v}_1 = \vec{v}_2$.

(4) If $h_1 \not\sim h_2$ and $\langle h_1, u_1\rangle, \langle h_2, u_2\rangle \in H$, then $\langle h_1, u_1\rangle \not\sim \langle h_2, u_2\rangle$.

Proof. Readily observed from the definition of $\sim$ in Definition 3.1. Q.E.D.

Example 3.3. As an illustration of modelling a Scotland Yard instance[6] as an extensive game with imperfect information, consider the digraph $G^\times = \langle V^\times, E^\times\rangle$, where

$$\begin{aligned} V^\times &= \{u_*, v_*, a, b, A, B, 1, 2, 3\} \\ E^\times &= \{\langle u_*, a\rangle, \langle u_*, b\rangle, \langle a, 1\rangle, \langle b, 2\rangle, \langle b, 3\rangle, \\ &\qquad \langle v_*, A\rangle, \langle v_*, B\rangle, \langle A, 1\rangle, \langle B, 2\rangle, \langle B, 3\rangle\}. \end{aligned}$$

For a depiction of $G^\times$, see Figure 2. Let $f^\times$ be an information function such that $f^\times(1) = \mathsf{hide}$ and $f^\times(2) = \mathsf{show}$. Conclude the construction of the Scotland Yard instance $\textsc{sy}^\times$, by putting u_* and v_* as the initial vertices of $\forall$ and $\exists$, respectively. In $\mathrm{SY}(\textsc{sy}^\times)$, the set of histories H contains exactly the following histories:

$\langle u_*, v_*\rangle$		
$\langle u_*, v_*, a\rangle$	$\langle u_*, v_*, a, B, 1\rangle$	$\langle u_*, v_*, a, B, 1, 3\rangle$
$\langle u_*, v_*, b\rangle$	$\langle u_*, v_*, b, A, 2\rangle$	$\langle u_*, v_*, b, A, 2, 1\rangle$
$\langle u_*, v_*, a, A\rangle$	$\langle u_*, v_*, b, A, 3\rangle$	$\langle u_*, v_*, b, A, 3, 1\rangle$
$\langle u_*, v_*, a, B\rangle$	$\langle u_*, v_*, b, B, 2\rangle$	$\langle u_*, v_*, b, B, 2, 2\rangle$!
$\langle u_*, v_*, b, A\rangle$	$\langle u_*, v_*, b, B, 3\rangle$	$\langle u_*, v_*, b, B, 2, 3\rangle$
$\langle u_*, v_*, b, B\rangle$	$\langle u_*, v_*, a, A, 1, 1\rangle$!	$\langle u_*, v_*, b, B, 3, 2\rangle$
$\langle u_*, v_*, a, A, 1\rangle$	$\langle u_*, v_*, a, B, 1, 2\rangle$	$\langle u_*, v_*, b, B, 3, 3\rangle$!

[6] The digraph under consideration does not have out-degree ≥ 1. The game terminates after two rounds, so adding reflexive edges on the vertices 1, 2, and 3 goes without affecting the winning conditions of the game.

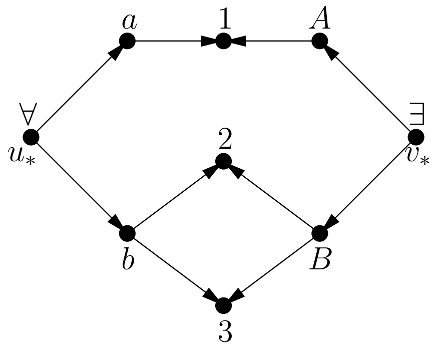


FIGURE 2. The digraph $G^{\times}$, allowing for a two-round Scotland Yard game.

The terminal histories marked with an exclamation mark are winning histories for $\exists$. Because $f^{\times}(1) = \mathsf{hide}$, the game at hand is a genuine game of imperfect information. This fact is reflected in the set of information cells $\mathcal{H}$, containing the following three non-singletons:

$$\begin{gathered}
\{\langle u_*, v_*, a\rangle, \langle u_*, v_*, b\rangle\}, \\
\{\langle u_*, v_*, a, A\rangle, \langle u_*, v_*, b, A\rangle\}, \\
\{\langle u_*, v_*, a, B\rangle, \langle u_*, v_*, b, B\rangle\}.
\end{gathered}$$

(Note that under the customary definition of $\sim$, one would not have the latter two information cells, as they belong to $\forall$.) Game theorists often find it convenient to present extensive games as trees, as in Figure 3.

4 A perfect information Scotland Yard game

We observed that Scotland Yard is a game with imperfect information and in Definition 3.1, we modeled it as an extensive game with imperfect information. This model one may find Scotland Yard's canonical means of analysis, for admittedly, it gives a natural account of the imperfect information that makes Scotland Yard such a fun game to play. Canonical or not, this does not imply, of course, that Scotland Yard can *only* be analyzed as an imperfect information game. In the remainder of this section we shall show how a Scotland Yard instance may also give rise to a game of perfect information. The underlying idea has it that during rounds in which $\forall$ moves and hides his whereabouts, he now picks up a *set of vertices* that contains all vertices where he can possibly be, from the viewpoint of $\exists$. In case $\forall$ has to show himself, he selects one vertex from the current set of vertices and announces this vertex as his new location.

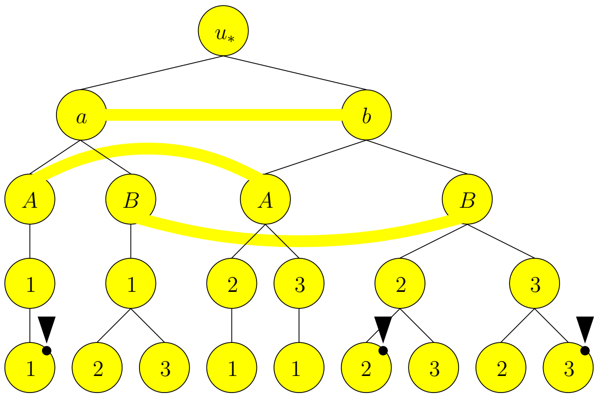


FIGURE 3. A graphical representation of the Scotland Yard game played on the game board constituted by the digraph $G^{\times}$ from Figure 2. A path from the root to any of its nodes represents a history in the game. For instance, the path $u_*, b, A, 2$ corresponds with the history $\langle u_*, v_*, b, A, 2\rangle$. The information cells are indicated by the shaded areas. The cells marked with an exclamation mark are won by $\exists$.

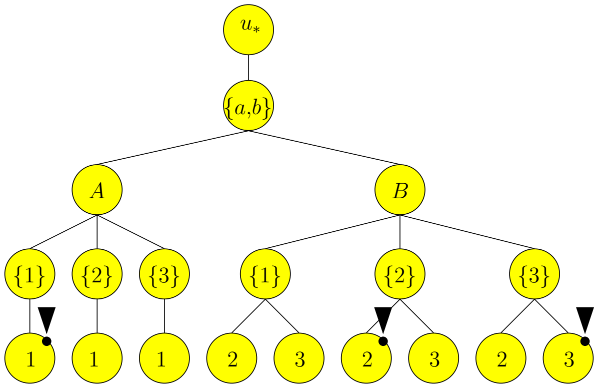


FIGURE 4. A graphical representation of the Scotland Yard-PI game played on the game board constituted by the digraph $G^{\times}$ from Figure 2. A path from the root to any of its nodes represents a history in the game. For instance, the path $u_*, \{a, b\}, A, \{2\}$ corresponds with the history $\langle\{u_*\}, v_*, \{a, b\}, A, \{2\}\rangle$. The cells marked with an exclamation mark are won by $\exists$.

More abstractly, $\forall$'s powers are lifted from the level of picking up vertices to the level of picking up sets of vertices. $\exists$'s powers remain unaltered, as compared to the game with imperfect information that was explicated above.

Modelling imperfect information by means of a power set construction—as we are about to do—is by no means new. For instance, the reader may find this idea occurring in the computational analyses of games with imperfect information [Re$_1$84, Pe$_3$AzRe$_1$01]. In logic, the idea of evaluating an Independence-friendly logic-sentence with respect to a set of assignments underlies Hodges' trump semantics [Ho$_1$97a]. In automata theory, the move to power sets is made when converting a non-deterministic finite automaton into a deterministic one, see [Ho$_4$Ul79].

As regards every single one of these disciplines, however, observe that the object that was analyzed through power sets is substantially more complex/powerful/bigger than the original object. For instance, in [Pe$_3$AzRe$_1$01] it was shown that three-player games with imperfect information can be undecidable. In the realm of IF logic it was proven [Ca$_1$Ho$_1$01] that no compositional semantics can be given based on single assignments only. It is well known that in the worst case converting a non-deterministic finite automaton makes the number of states increase exponentially.

In view of these results it is striking that one can define a Scotland Yard game with perfect information using a power set argument, without experiencing a combinatorial explosion, cf. Theorem 6.3. What is meant by "highly similar" is made precise in Section 5. First let me postulate the game rules for the Scotland Yard game with perfect information and define its extensive game form in Definition 4.1.

Let $\textsc{sy} = \langle G, \langle u_*, \vec{v}_* \rangle, f \rangle$ be a Scotland Yard instance as in Definition 1.1. The initial position of the Scotland Yard-PI game constituted by $\textsc{sy}$ is similar to the initial position of the Scotland Yard game that is constituted by $\textsc{sy}$. That is, a $\forall$ pawn is positioned on u_* and for every $1 \leq j \leq n$, the $\exists_j$ pawn is positioned on $\vec{v}(j)$. In Scotland Yard-PI, $\forall$ does not have one pawn at his disposal but as many as there are vertices in G.

Fix $i = 1$, $U = \{u_*\}$, and $\vec{v} = \vec{v}_*$; round i of Scotland Yard-PI goes as follows:

1-PI. If $U - \{\vec{v}\} = \varnothing$, then the game stops and $\exists$ wins. If $U - \{\vec{v}\} \neq \varnothing$ and $i > k$ the game also stops but $\exists$ loses.

2-PI. Let $U' = E(U - \{\vec{v}\})$. If $f(i) = \mathsf{hide}$, then set $U = U'$ and $\forall$ positions a $\forall$ pawn on every vertex v in U. If $f(i) = \mathsf{show}$, then $\forall$ picks a vertex $u' \in U'$, removes all his pawns from the board, and puts one pawn on u'. Set $U = \{u'\}$.

3-PI. Player $\exists$ chooses a vector $\vec{v}' \in V^n$, such that $E(\vec{v}, \vec{v}')$, and for every $1 \leq j \leq n$, moves pawn $\exists_j$ to $\vec{v}'(j)$. Set $\vec{v} = \vec{v}'$.

4-PI. Set $i = i + 1$.

Clearly, for arbitrary SY, the Scotland Yard-PI game constituted by SY is a game of perfect information. Thus extensive game theory provides natural means of analysis.

Definition 4.1. Let SY $= \langle G, \langle u_*, \vec{v}_* \rangle, f \rangle$ be a Scotland Yard instance. Then, let the *extensive Scotland Yard-PI game constituted by* SY be defined as the tuple SY-PI(SY) $= \langle N_{\mathrm{PI}}, H_{\mathrm{PI}}, P_{\mathrm{PI}}, U_{\mathrm{PI}} \rangle$, where

- $N_{\mathrm{PI}} = \{\exists, \forall\}$.
- H_{PI} is the smallest set containing the strings $\langle \{u_*\} \rangle, \langle \{u_*\}, \vec{v}_* \rangle$, that is closed under taking actions for $\exists$ and $\forall$:
  - If $\langle h, U, \vec{v} \rangle \in H_{\mathrm{PI}}$, $\ell(\langle h, U, \vec{v} \rangle) < k$, $f(\ell(\langle h, U, \vec{v} \rangle) + 1) = \mathsf{hide}$, and $U - \{\vec{v}\} \neq \varnothing$, then $\langle h, U, \vec{v}, E(U - \{\vec{v}\}) \rangle \in H_{\mathrm{PI}}$.
  - If $\langle h, U, \vec{v} \rangle \in H_{\mathrm{PI}}$, $\ell(\langle h, U, \vec{v} \rangle) < k$, and $f(\ell(\langle h, U, \vec{v} \rangle) + 1) = \mathsf{show}$, then $\{\langle h, U, \vec{v}, \{u'\} \rangle \mid u' \in E(U - \{\vec{v}\})\} \subseteq H_{\mathrm{PI}}$.
  - If $\langle h, U, \vec{v}, U' \rangle \in H_{\mathrm{PI}}$ and $E(\vec{v}, \vec{v}')$, then $\langle h, U, \vec{v}, U', \vec{v}' \rangle \in H_{\mathrm{PI}}$.

  Let $\prec_{\mathrm{PI}}$ be the *immediate successor* relation on H_{PI}. That is, the smallest relation closed under the following conditions:

  - If $h, \langle h, U \rangle \in H_{\mathrm{PI}}$, then $h \prec_{\mathrm{PI}} \langle h, U \rangle$.
  - If $\langle h, U \rangle, \langle h, U, \vec{v} \rangle \in H_{\mathrm{PI}}$, then $\langle h, U \rangle \prec_{\mathrm{PI}} \langle h, U, \vec{v} \rangle$.
- Again, the value of P is readily determined from the history's form, in the sense that $P(\langle h, U \rangle) = \exists$ and $P(\langle h, U, \vec{v} \rangle) = \forall$, no matter h, U, and $\vec{v}$.
- Naturally,

$$U(\langle h, U, \vec{v} \rangle) = \begin{cases} \mathsf{win} & \text{if } U - \{\vec{v}\} = \varnothing \\ \mathsf{lose} & \text{if } U - \{\vec{v}\} \neq \varnothing. \end{cases}$$

These definitions may be appreciated best by checking SY-PI(SY$^\times$), where SY$^\times$ $= \langle G^\times, \langle u_*, \vec{v}_* \rangle, f^\times \rangle$ and $G^\times$ is the digraph depicted in Figure 2. We skip writing down all histories in this particular game, leaving the reader with a graphical representation of its game tree in Figure 4.

5 An effective equivalence

In this section, the similarity between Scotland Yard and its perfect information variant is established. Making use of this similarity, we prove in Theorem 5.7 that for any instance SY, $\exists$ has a winning strategy in SY(SY) iff she has one in SY-PI(SY). In order to prove this result it will be shown in Lemma 5.6 that the structures $\langle \mathcal{H}, \prec \rangle$ and $\langle H_{\mathrm{PI}}, \prec_{\mathrm{PI}} \rangle$ are isomorphic, in virtue of the bijection β.

Main result of this subsection resides in Lemma 5.6, saying that the structures $\langle \mathcal{H}, \prec \rangle$ and $\langle H_{\mathrm{PI}}, \prec_{\mathrm{PI}} \rangle$ are isomorphic. The witness of this isomorphism is the bijection β, shortly defined in Definition 5.1. The intermediate results can be proved by inductive arguments. As the proofs do not contribute too much to the content of this paper we decided to not incorporate them. The reader can find a detailed account of these proofs in [Se$_2$06].

The function β is a map from histories in the perfect information game SY-PI(SY) to information cells in the game SY(SY). An information cell is a set of histories that cannot be distinguished by $\exists$. The perfect information Scotland Yard game was defined in such a way that $\exists$'s imperfect information in SY(SY) is propagated to perfect information about sets in SY-PI(SY). It will be observed through the map β that there is a bijection between information cells—sets of histories—in the imperfect information game, and histories in the perfect information game that hold sets of vertices owned by $\forall$. Thus, β is a map from the information cells in SY(SY) to histories in SY-PI(SY).

Definition 5.1. Let SY(SY) and SY-PI(SY) be games constituted by SY. Define the function $\beta : H_{\mathrm{PI}} \to \wp(H)$ inductively as follows:

$$\begin{array}{rcl}
\beta(\langle\{u_*\}\rangle) & = & \{\langle u_*\rangle\} \\
\beta(\langle\{u_*\}, \vec{v}_*\rangle) & = & \{\langle u_*, \vec{v}_*\rangle\} \\
\beta(\langle h, U\rangle) & = & \{\langle g, u\rangle \in H \mid g \in \beta(h), u \in U\} \\
\beta(\langle h, U, \vec{v}\rangle) & = & \{\langle g, u, \vec{v}\rangle \in H \mid \langle g, u\rangle \in \beta(\langle h, U\rangle)\}.
\end{array}$$

The function β is (partially) depicted in Figure 5 mapping the histories from SY-PI(SY$^\times$) to sets of histories from SY(SY$^\times$).

Proposition 5.2 states that if in a history $h \in H_{\mathrm{PI}}$ a pawn (owned by either player) is positioned on a vertex, then also in $\beta(h)$ there exists a history in which this vertex is occupied by a pawn.

Proposition 5.2. For every history $h' \in H_{\mathrm{PI}}$, the following hold:

(1) If $h' = \langle h, U\rangle$ and $f(\ell(\langle h, U\rangle)) = \mathsf{hide}$, then it is the case that $U = \{u \mid \langle g, u\rangle \in H$, for some $g \in \beta(h)\}$.

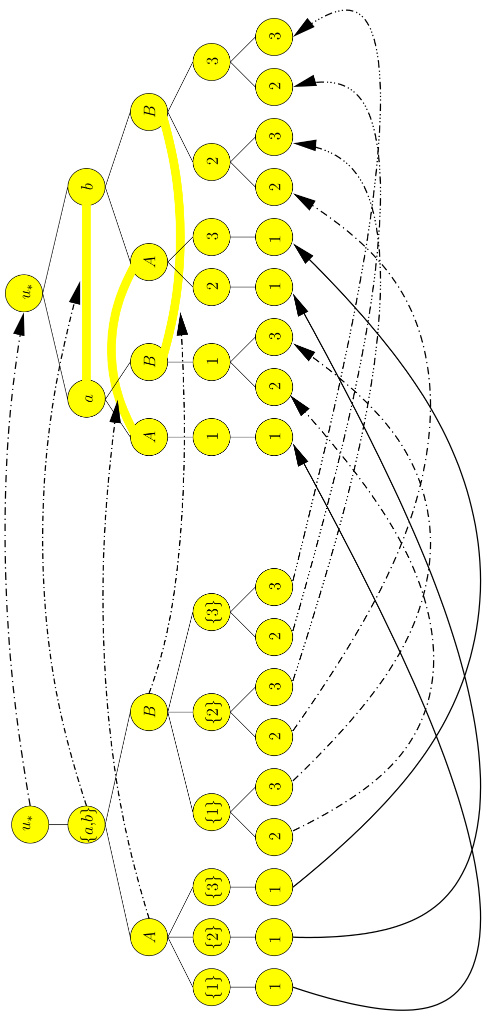


FIGURE 5. A partial depiction of the bijection β from histories in SY-PI(SY$^\times$) to sets of histories from SY(SY$^\times$). β is displayed using several kinds of arrows to enhance readability. Note that this visualization does not reflect any conceptual difference. Sets of histories in the range of β (found in the right-hand structure) turn out to be information cells, cf. Lemma 5.5.

(2) If $P(h') = \forall$ and $f(\ell(h')+1) = \mathsf{show}$, then it is the case that $\{u \mid h' \prec \langle h', \{u\}\rangle$, for some $\langle h', \{u\}\rangle \in H_{\mathrm{PI}}\} = \{u \mid \langle g, u\rangle \in H$, for some $g \in \beta(h')\}$.

(3) If $h' = \langle h, U\rangle \in H_{\mathrm{PI}}$ and $u \in U$, then there exists a history $g \in \beta(h)$ such that $\langle g, u\rangle \in H$.

(4) If $h' = \langle h, U, \vec{v}\rangle \in H_{\mathrm{PI}}$, then it is the case that $\beta(\langle h, U, \vec{v}\rangle)$ equals $\{\langle g, u, \vec{v}\rangle \mid \langle g, u\rangle \in \beta(\langle h, U\rangle)\}$.

Proposition 5.3 is the converse of the previous proposition, as it links up histories in H with histories in H_{PI}.

Proposition 5.3. For every $g' \in H$, the following hold:

(1) If $g' = \langle g, u\rangle \in H$, then there exists a $\langle g, U\rangle \in H_{\mathrm{PI}}$ such that $g \in \beta(h)$ and $u \in U$.

(2) If $g' = \langle g, u, \vec{v}\rangle \in H$, then there exists a $\langle h, U, \vec{v}'\rangle \in H_{\mathrm{PI}}$ such that $\langle g, u\rangle \in \beta(\langle h, U\rangle)$ and $\vec{v} = \vec{v}'$.

For β to be bijection between H_{PI} and $\mathcal{H}$, it ought to be the case that β has range $\mathcal{H}$ rather than $\wp(H)$. We lay down the following result.

Lemma 5.4. β is a function of type $H_{\mathrm{PI}} \to \mathcal{H}$.

The latter lemma is strengthened in the following lemma.

Lemma 5.5. β is a bijection between H_{PI} and $\mathcal{H}$.

The isomorphism result follows from tying together the previous statements.

Lemma 5.6. The structures $\langle H_{\mathrm{PI}}, \prec_{\mathrm{PI}}\rangle$ and $\langle \mathcal{H}, \prec\rangle$ are isomorphic.

Proof. Lemma 5.5 showed that β is a bijection between H_{PI} and $\mathcal{H}$. It remains to be shown that β preserves structure, that is, for every pair of histories $h, h' \in H_{\mathrm{PI}}$, it is the case that $h \prec_{\mathrm{PI}} h'$ iff $\beta(h) \prec \beta(h')$. Recall that for $C' \in \mathcal{H}$ to be the immediate successor of $C \in \mathcal{H}$, there must exist two histories g, g' from C, C', respectively, such that $g \prec g'$. The claim is proved by a straightforward inductive argument on the length of the histories in H_{PI}. We shall omit spelling out the details of the proof, only mentioning the Propositions on which it relies:

From left to right. Follows from Proposition 5.2 (3) and (4).

From right to left. Follows from Proposition 5.3 (1) and (2). Q.E.D.

Scotland Yard and its perfect information variant are highly similar in the sense that the game trees to which the games give rise are isomorphic.

The structures $\langle H_{\mathrm{PI}}, \prec_{\mathrm{PI}} \rangle$ and $\langle \mathcal{H}, \prec \rangle$ are not only isomorphic, they also preserve the property of being winnable for the cops. To flesh out this claim, let $f : B \to C$ and $g : A \to B$ be functions. Then, $f \cdot g$ denotes the function of type $A \to C$ such that for every $a \in A$, $f \cdot g(a) = f(g(a))$.

Making use of the fact that $\langle H_{\mathrm{PI}}, \prec_{\mathrm{PI}} \rangle$ and $\langle \mathcal{H}, \prec \rangle$ are isomorphic, an inductive argument proves that S is a winning strategy in SY-PI(sy) iff $S \cdot \beta$ is a winning strategy in SY(SY).[7] Thus, we arrive at the following result.

Theorem 5.7. Let SY be a Scotland Yard instance. Then, $\exists$ has a winning strategy in SY(SY) iff she has a winning strategy in SY-PI(SY).

6 Scotland Yard is PSPACE-complete

In this section, we define Scotland Yard as a decision problem and prove that it is **PSPACE**-complete, both the perfect and the imperfect information game. Hence, the imperfect information in Scotland Yard does not increase the game's complexity, under the current analysis.

Let SCOTLAND YARD be the set of all Scotland Yard instances SY such that $\exists$ has a winning strategy in SY(SY). As a special case let the set of Scotland Yard instances SCOTLAND YARD$_{\clubsuit}$ equal

$$\{\langle G, \langle u_*, \vec{v}_* \rangle, f \rangle \in \textsc{Scotland Yard} \mid f \text{ has range } \{\clubsuit\}\},$$

where $\clubsuit \in \{\mathsf{show}, \mathsf{hide}\}$.

In this section, we show that SCOTLAND YARD and SCOTLAND YARD$_{\mathsf{show}}$ both are complete for **PSPACE**. From this one may conclude that the imperfect information in Scotland Yard does not have a computational impact. Surprisingly, if $\exists$ cannot see the whereabouts of $\forall$ at any stage of the game, the decidability problem ends up being **NP**-complete. That is, SCOTLAND YARD$_{\mathsf{hide}}$ is complete for **NP**. The latter claim is substantiated in Section 7.

Lemma 6.1. SCOTLAND YARD $\in$ **PSPACE**.

Proof. Required is a **PSPACE** algorithm that for arbitrary Scotland Yard instances SY decides whether $\exists$ has a winning strategy in SY(SY). By Theorem 5.7, it is sufficient to provide a **PSPACE** algorithm that decides the same problem with respect to SY-PI(SY). This equivalence comes in useful, because SY-PI(SY) is a game of perfect information and can for this reason be dealt with by means of the traditional machinery. In fact, the very same

[7] One of the anonymous referees made the valuable point that winning strategy preservation follows almost directly from the fact that the two structures are isomorphic.

machinery supplied by Papadimitriou cited in Section 1 will do. Recall that Papadimitriou, namely, observed that deciding the value of a game with perfect information can be done in **PSPACE** if the following requirements are met:

- the length of any legal sequence of moves is bounded by a polynomial in the size of the input; and
- given a "board position" of the game there is a polynomial-space algorithm which constructs all possible subsequent actions and board positions; or, if there aren't any, decides whether the board position is a win for either player.

It is easy to see that SY-PI(SY) meets those conditions. As to the first one, namely, the length of the description of any history is polynomially bounded by the number of rounds k of the game. By assumption $k \leq \|V\| \leq \|\mathrm{SY}\|$, whence the description of every history is polynomial in the size of the input. As regards the second condition, if $\langle h, U, \vec{v}\rangle$ is a non-terminal history, then its successors are either (depending on f) only $\langle h, U, \vec{v}, \{w_1, \ldots, w_m\}\rangle$ or all of $\langle h, U, \vec{v}, \{w_1\}\rangle, \ldots, \langle h, U, \vec{v}, \{w_m\}\rangle$, where $E(U - \{\vec{v}\}) = \{w_1, \ldots, w_m\}$. Those can clearly be constructed in **PSPACE**.

In the worst case, for an arbitrary history $\langle h, U, \vec{v}, U'\rangle$ owned by $\exists$ there are $\|V\|^n$ many vectors $\vec{v}'$ such that $E(\vec{v}, \vec{v}')$, where n is the number of $\exists$'s pawns on the game board. This number is exponential in the size of the input. Nevertheless, every vector $\vec{v}'$ in $V^n = \{v_1, \ldots, v_{\|V\|}\}^n$ can be constructed in polynomial space, simply by writing down the vector $\langle v_1, \ldots, v_1\rangle \in V^n$ that comes first in the lexicographical ordering and successively constructing the remaining vectors that follow it up in the same ordering. Q.E.D.

Hardness is shown by reducing from QBF.

Lemma 6.2. SCOTLAND YARD$_{\mathsf{show}}$ is **PSPACE**-hard.

Proof. Given a QBF instance $\psi = \forall x_1 \exists y_1 \ldots \forall x_n \exists y_n\ \varphi$, where $\varphi = C_1 \wedge \ldots \wedge C_m$ is a boolean formula in 3-CNF, it suffices to construct in **P** a Scotland Yard instance SY_ψ, such that $\psi \in$ QBF if and only if $\mathrm{SY}_\psi \in$ SCOTLAND YARD$_{\mathsf{show}}$. To this end, let me construct the initial position of the game constituted by SY_ψ. The formal specification of SY_ψ follows directly from these building blocks.

Set $i = 0$. For $i \leq n + 1$, do as follows:

- If $i = 0$, lay down the *opening-gadget*, that is schematically depicted in Figure 6.a. Moreover, distribute the pawns from

$$\{\exists_{x_1}, \ldots, \exists_{x_n}, \exists_{y_1}, \ldots, \exists_{y_n} \exists_d, \forall\}$$

over the vertices of the opening-gadget as indicated in its depiction.

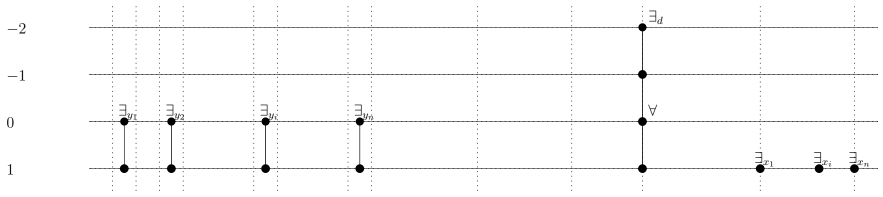
- If $1 \leq i \leq n$, first put the *x_i-gadget* at the bottom of the already constructed game board. Next, put the *y_i-gadget* below the justly introduced x_i-gadget. Figures 6.b and 6.c give a schematic account of the x_i-gadget and y_i-gadget, respectively. (Note that as a result of these actions, every vertex in the board game is connected to at least one other vertex, except for the ones on the top row of the opening-gadget and the ones on the bottom row of the y_i-gadget.)
- If $i = n + 1$, put the *clause-gadget* (see Figure 6.d) at the bottom of the already constructed board game. This gadget requires a little tinkering before construction terminates, in order to encode the boolean formula φ by adding edges to the clause-gadgets (not present in the depiction), as follows:
  - For every variable $z \in \{\vec{x}, \vec{y}\}$ and clause C in φ: If z occurs as a literal in C, then join the vertices named "$+z$" and "C" by an edge. If $\neg z$ occurs as a literal in C, then join the vertices named "$-z$" and "C" by an edge.
- Set $i = i + 1$.

Note that the board game can be considered to consists of *layers*, that are indicated by the horizontal, dotted lines. These layers are numbered $-2, -1, \ldots, 4n + 5$, enabling us to refer to these layers when describing strategies. Note that the division in layers is not complete: in between layer $4(i-1)+3$ and layer $4(i-1)+4$ of the x_i-gadget are two floating vertices.

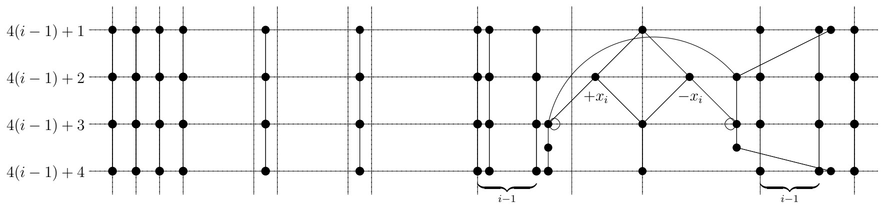
The formal specifications of the graph and the initial positions of SY are easily derived from the previous descriptions. Therefore, SY_ψ is fully specified after putting $f : \{1, \ldots, 4n + 5\} \to \{\mathsf{show}\}$. Hence, SY_ψ is an instance of SCOTLAND YARD$_{\mathsf{show}}$.

It remains to be shown that $\psi \in$ QBF iff $\text{SY}_\psi \in$ SCOTLAND YARD$_{\mathsf{show}}$.

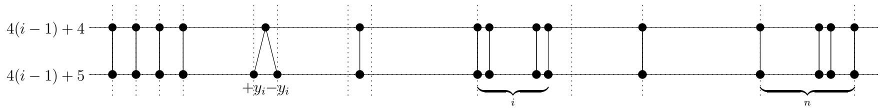
From left to right. Suppose $\{\mathsf{true}, \mathsf{false}\} \models \psi$, then there is a *way of subsequently picking* truth values for the existentially quantified variables that renders ψ's boolean part φ true, no matter what truth values are assigned to the universally quantified variables. $\exists$'s winning strategy in $\text{SY}(\text{SY}_\psi)$ (being witness of the fact that $\text{SY}_\psi \in$ SCOTLAND YARD$_{\mathsf{show}}$) shall be read off from the aforementioned way of picking. We do so by interpreting moves in $\text{SY}(\text{SY}_\psi)$ as assigning truth values to variables and vice versa: Actions performed by $\forall$ from layer $4(i-1)+1$ to layer $4(i-1)+2$ will be interpreted as assigning a truth value to the universally quantified variable x_i. In particular, a move by $\forall$ to the vertex named "$+x_i$" ("$-x_i$") will



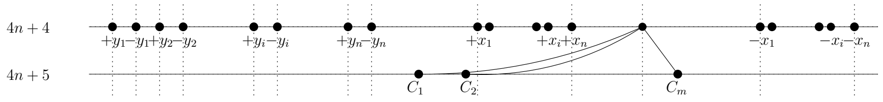
(a) Opening-gadget



(b) x_i-gadget



(c) y_i-gadget



(d) Clause-gadget

FIGURE 6. The gadgets that make up the initial position of the board game constituted by SY(SY_ψ). The dotted lines are merely "decoration" of the game board, to enhance readability. The horizontal, dotted lines are referred to as "layers".

be interpreted as assigning to x_i the value true (false). Conversely, if $\exists$'s way of picking prescribes assigning true (false) to y_i this will be reflected in SY(SY_ψ) by moving $\exists_{y_i}$ to the vertex named "$+y_i$" ("$-y_i$") on layer $4(i-1)+5$.

Roughly speaking, $\exists$ goes about as follows: when she is to chose between moving $\exists_{y_i}$ to the vertex named "$+y_i$" or "$-y_i$" she understands $\forall$'s previous moves as a truth assignment and observes which truth value is prescribed by the way of picking. Next, she interprets this truth value as a move in SY(SY_ψ) as described above and moves $\exists_{y_i}$ to the according vertex. This intuition underlies the full specification of $\exists$'s strategy, described below:

For $0 \leq i \leq n+1$ let $\exists$'s strategy be as follows:

- Above all: If any pawn can capture $\forall$, do so!
- For every pawn that stands on a vertex on layer j that is connected to exactly one vertex on layer $j+1$, move it to this vertex. If the pawn at stake is actually $\exists_{x_i}$ standing on a vertex on layer $4(i-1)+3$, it cannot move to the vertex on layer $4(i-1)+4$, because there is a vertex v in between. In this case, move $\exists_{x_i}$ to v and on the next round of the game move it downwards to layer $4(i-1)+4$.
- If $\exists_{x_i}$ stands on a vertex on layer $4(i-1)+2$, then move it to the vertex on layer $4(i-1)+3$ that is connected to the vertex where $\forall$ is positioned.
- If $\exists_{y_i}$ stands on a vertex on layer $4(i-1)+4$, and the way of picking prescribes assigning true (false) to y_i, then move it to the vertex on layer $4(i-1)+5$ that is named "$+y_i$" ("$-y_i$").
- If $\exists_d$ stands on a vertex on layer j that has two successors on layer $j+1$, then move it along the left-hand (right-hand) edge, if j is even (odd).
- If $\exists_z$ (for $z \in \{\vec{x}, \vec{y}\}$) stands on a vertex on layer $4n+4$ and this vertex is not connected to a vertex on which $\forall$ is positioned, move it along an arbitrary edge (possibly upwards).

As to $\forall$'s behavior we claim without rigorous proof that after $4n+4$ rounds of the game (that is, without being captured at an earlier stage of the game) he has traversed a path leading through exactly one of the vertices named "$+x_i$" and "$-x_i$", for every $x_i \in \{\vec{x}\}$, ending up in a vertex named "C", for some clause C in φ. To see that this must be the case: moving $\forall$ upwards at any stage of the game results in an immediate capture by $\exists_d$. (In fact, $\exists_d$'s sole purpose in life is capturing $\forall$, if he moves upward.) If $\forall$

is moved to one of the reflexive vertices on layer $4(i-1)+3$ he is captured by $\exists_{x_i}$ who moves along the reflexive edge.

Upon arriving at layer $4n+4$, pawn $\exists_z$ (for $z \in \{\vec{x}, \vec{y}\}$) stands on a vertex named "$+z$" or "$-z$", reflecting that z was assigned true or false, respectively. By assumption on the successfulness of the way of picking, that guided $\exists$ through $\mathrm{SY}(\textsc{sy}_\psi)$, it is the case that the truth assignment that is associated with the positions of the pawns $\exists_{x_1}, \ldots, \exists_{x_n}, \exists_{y_1}, \ldots, \exists_{y_n}$ makes φ true. That is, under that truth assignment, for every clause C in φ there is a literal L that is made true. Now, if $L = z$, then $\exists_z$ stands on the vertex named "$+z$" and this vertex and the vertex named "C" are joined by an edge; and if $L = \neg z$, then $\exists_z$ stands on the vertex named "$-z$" and this vertex and the vertex named "C" are joined by an edge. So no matter to which vertex named "C" pawn $\forall$ moves during his $4n+5$th move, for at least one $z \in \{\vec{x}, \vec{y}\}$ it is the case that $\exists_z$ can move to this vertex named "C" and capture him there.

From right to left. Suppose $\{\mathsf{true}, \mathsf{false}\} \not\models \psi$, then there is a way of picking truth values for the universally quantified variables that renders the boolean part false, no matter what truth values are subsequently assigned to the existentially quantified variables. We leave out the argumentation that this way of picking constitutes a winning strategy for $\forall$ in $\mathrm{SY}(\textsc{sy}_\psi)$, as it is similar to the argumentation in the converse direction. But note one crucial property of $\forall$'s winning strategy: it moves pawn $\forall$ downwards, during every round in the game. Therefore, the only round in which it can be captured is the last one: on a vertex on layer $4n+5$.

Close attention is required, though, to $\exists$'s behavior. That is, it is to be observed that $\exists$ cannot change her sad destiny (losing) by deviating from the behavior specified in the rules below. The gist of this behavior is that it results in pawn $\exists_{x_i}$ remembering $\forall$'s moves on layer $4(i-1)+1$ and that after $4n+4$ rounds the pawns $\exists_{x_1}, \ldots, \exists_{x_n}, \exists_{y_1}, \ldots, \exists_{y_n}$ all stand on a vertex on layer $4n+4$. The point is that just as above, the positions of these pawns on vertices on layer $4n+4$ reflect a truth assignment. This time however, the truth assignment falsifies the boolean part φ.

The rules are as follows:

(1) If $\exists_{x_i}$ stands on a vertex on layer $4(i-1)+2$, then move it to the vertex on layer $4(i-1)+3$ that is connected to the vertex where $\forall$ is positioned.

(2) If $\exists_d$ stands on a vertex on layer j that is connected to two vertices below, then move it along the left-hand (right-hand) edge, if j is even (odd); or along the right-hand (left-hand) edge, if j is even (odd).

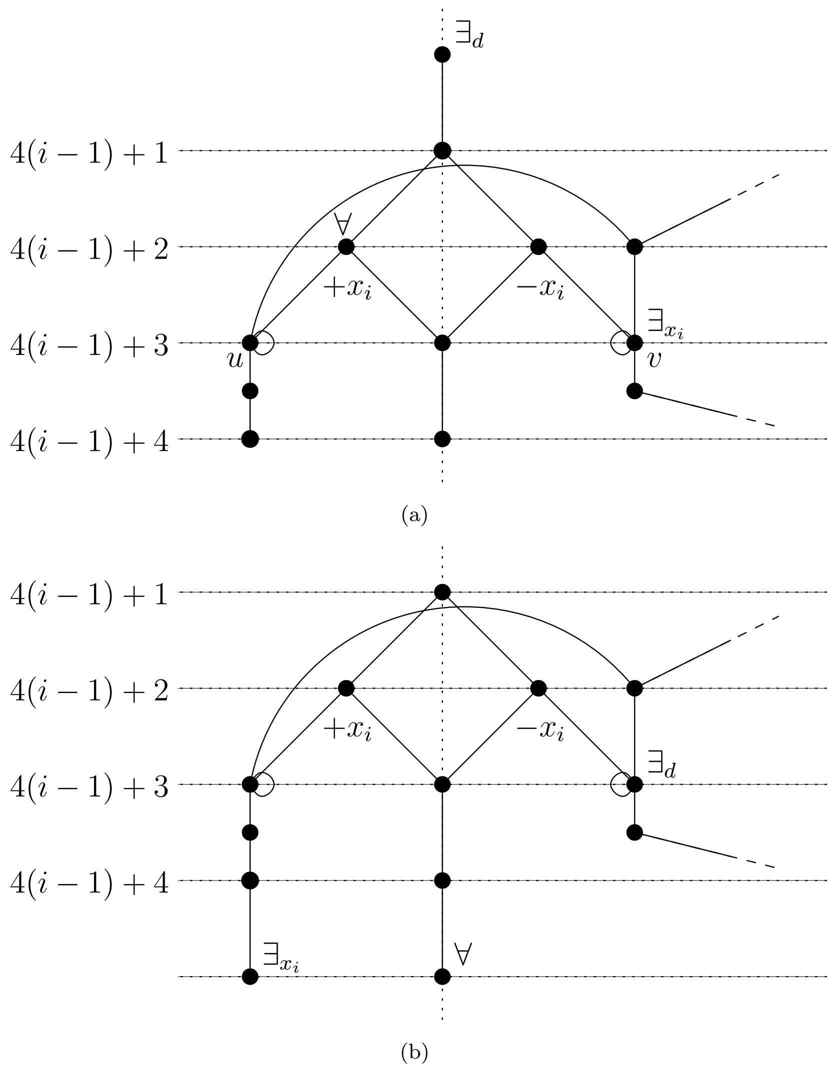


FIGURE 7. Positions on the game board that may occur if $\exists$ does not play according to rule (1) and (2), depicted in (a) and (b), respectively.

(3) For every pawn that stands on a vertex on layer j that is connected to exactly one vertex on layer $j+1$, move it to this vertex. (With the same exception as before with regard to $\exists_{x_i}$ standing on a vertex on layer $4(i-1)+3$.)

We argue that not behaving in correspondence with (1)-(3) will also result in a loss for $\exists$:

(1) Suppose $\exists_{x_i}$ stands on the vertex on layer $4(i-1)+2$, with two options: u and v. Let u be the vertex on layer $4(i-1)+3$ that is connected to the vertex where $\forall$ is positioned (see Figure 7.a). For the sake of the argument let us suppose that $\exists_{x_i}$ is moved to v, violating rule (1). In that case, $\forall$ may safely move to u. If $\forall$ continues the game by moving its pawn downwards it wins automatically, since after the final round (round $4n+5$) its pawn stands on a vertex on layer $4n+4$, due to the extra vertex sitting in between layers $4(i-1)+3$ and $4(i-1)+4$, without there being any opportunity for $\exists$ to capture him. As such, the pawn $\exists_{x_i}$ is forced to *remember* what vertex $\forall$ visited on layer $4(i-1)+2$: the one named "$+x_i$" or "$-x_i$".

(2) Suppose $\exists_d$ stands on a vertex on layer $4(i-1)+1$ and from there moves along the right-hand edge twice (see Figure 7.b). $\forall$ can exploit this move by moving as he would move normally, except for round $4n+5$, during which he moves upwards. This behavior results in a guaranteed win for $\forall$, since none of $\exists$'s pawns is pursuing $\forall$ closely enough to capture it, after moving upwards.

(3) Suppose any pawn controlled by $\exists$ moves upwards instead of downwards. This can never result in a win for $\exists$, because $\forall$ (behaving as he does) can only be captured in the last round of the game, on a vertex on layer $4n+5$. In particular, any pawn $\exists_z$, for $z \in \{\vec{x}, \vec{y}\}$, the shortest path to a vertex on layer $4n+5$ is of length $4n+5$. Now, if $\exists_z$ is moved upwards, it cannot (during the last round of the game) capture $\forall$.

This conclude the proof. Q.E.D.

The previous two lemmata are sufficient arguments to settle completeness.

Theorem 6.3. SCOTLAND YARD and SCOTLAND YARD$_{\mathsf{show}}$ are **PSPACE**-complete.

Proof. Lemma 6.1 holds that SCOTLAND YARD is solvable in **PSPACE**. To check whether an instance SY has a function f with range $\{\mathsf{show}\}$ is trivial, therefore, also SCOTLAND YARD$_{\mathsf{show}}$ is solvable in **PSPACE**.

PSPACE-hardness was proven for SCOTLAND YARD$_{\mathsf{show}}$ in Lemma 6.2. Since the latter problem is a specialization of SCOTLAND YARD, it follows immediately that SCOTLAND YARD is **PSPACE**-hard as well. Hence, both problems are complete for **PSPACE**. Q.E.D.

7 Ignorance is (computational) bliss

Intuitively, adding imperfect information makes a game harder. However, if one restricts oneself to Scotland Yard instances in which $\forall$'s whereabouts are only known at the beginning of the game, then *deciding* whether $\exists$ has a winning strategy is **NP**-complete, cf. Theorem 7.3. After the proof of this theorem, we argue that from a quantitative point of view it is indeed harder for $\exists$ to win an arbitrary Scotland Yard game, thus backing up our pre-computational intuitions.

Lemma 7.1. SCOTLAND YARD$_{\mathsf{hide}}$ $\in$ **NP**.

Proof. We make use of the equivalence between the Scotland Yard game and its perfect information counterpart Scotland Yard-PI. It suffices to give an **NP** algorithm that decides whether $\exists$ has a winning strategy in an arbitrary SY-PI(SY), where SY's information function has range $\{\mathsf{hide}\}$. That is, for every integer i on which f is properly defined, we have that $f(i) = \mathsf{hide}$. Let me now repeat the game rule from page 224 that regulates $\forall$'s behavior in the game of Scotland Yard-PI:

2-PI. Let $U' = E(U - \{\vec{v}\})$. If $f(i) = \mathsf{hide}$, then set $U = U'$ and $\forall$ positions a $\forall$ pawn on every vertex v iff $v \in U$. If $f(i) = \mathsf{show}$, then $\forall$ picks a vertex $u' \in U'$, removes all his pawns from the board, and puts one pawn on u'. Set $U = \{u'\}$.

Since for no $1 \leq i \leq k$, $f(i)$ equals show, we can harmlessly replace it by the following rule:

2-PI$'$. Set $U' = E(U - \{\vec{v}\})$ and $\forall$ positions a $\forall$ pawn on every vertex v iff $v \in U$.

Doing so yields a game in which $\forall$ plays no active role anymore, in the sense that the set U at any round of the game is completely determined by $\exists$'s past moves. Put differently, any game constituted by an instance of SCOTLAND YARD$_{\mathsf{hide}}$ is essentially a one-player game! Having obtained this insight, it is easy to see that the following algorithm decides in non-deterministic polynomial time whether $\exists$ has a winning strategy in the k-round SY-PI(SY):

- Non-deterministically guess a k number of n-dimensional vectors of vertices $\vec{v}_1, \ldots, \vec{v}_k \in V^n$.

- Set $U = \{u_*\}$, $\vec{v} = \vec{v}_*$, and $i = 1$; then for $i \leq k$ proceed as follows:
  - If $E(\vec{v}, \vec{v}_i)$, then set $\vec{v} = \vec{v}_i$; else, reject.
  - If $(U - \{\vec{v}\}) = \varnothing$, then accept; else, set $U = (U - \{\vec{v}\})$.
  - Set $i = i + 1$.
- If after k rounds there are still $\forall$ pawns present on the game board, reject.

This algorithm is correct: $\exists$ has a winning strategy in SY-PI(SY) iff it accepts SY. Hence, SCOTLAND YARD$_{\mathsf{hide}}$ is in **NP**. Q.E.D.

To prove hardness, we reduce from 3-SAT, assuming that no clause in a 3-SAT instance contains two copies of one propositional variable. This goes without loss of generality.

Lemma 7.2. SCOTLAND YARD$_{\mathsf{hide}}$ is **NP**-hard.

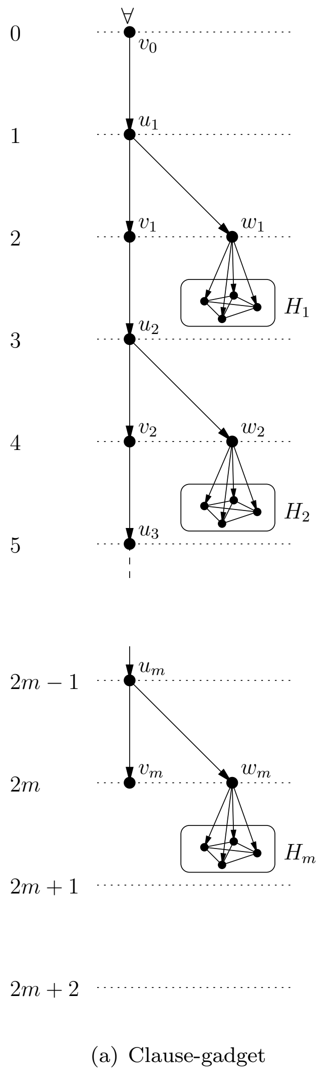
Proof. To reduce from 3-SAT, let $\varphi = C_1 \wedge \ldots \wedge C_m$ be an instance of 3-SAT over the variables $x_1, \ldots, x_n$. On the basis of φ we shall construe a Scotland Yard instance SY$_\varphi$ such that φ is satisfiable iff $\exists$ has a winning strategy in SY-PI(SY$_\varphi$). In fact, SY$_\varphi$ will be read off from the initial game board that is put together as follows.

Set $i = 0$; for $i \leq n$ proceed as follows:

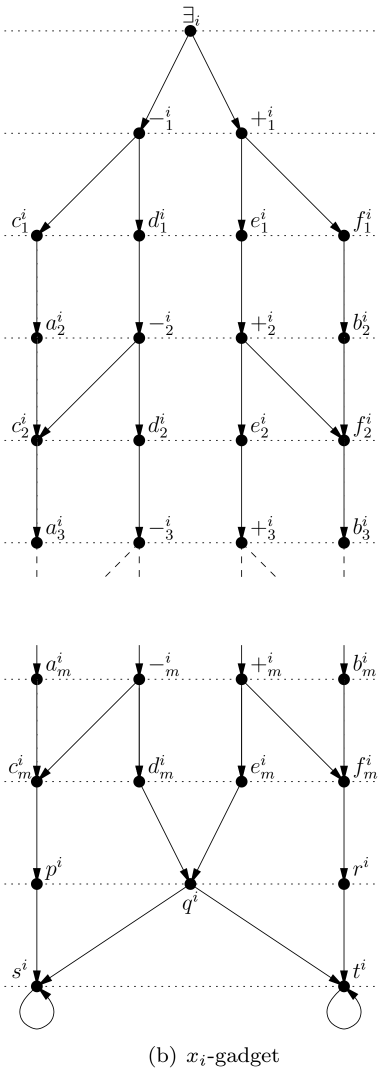
- If $i = 0$, lay down the *clause-gadget* from Figure 8.a. The sub-graphs H_j are fully connected graphs with four elements, whose vertices are connected with the vertices w_j, for $1 \leq j \leq m$.
- If $1 \leq i \leq n$, put the x_i*-gadget* to the right of the already constructed game board, see Figure 8.b. It will be convenient to refer to the vertex q^i by means of $-^i_{m+1}$ and $+^i_{m+1}$.

  For every $0 \leq j \leq m$, do as follows:
  - If x_i occurs as a literal in C_j, add the edges $\langle +^i_j, w_j \rangle$ and $\langle w_j, +^i_{j+1} \rangle$.
  - If $\neg x_i$ occurs as a literal in C_j, add the edges $\langle -^i_j, w_j \rangle$ and $\langle w_j, -^i_{j+1} \rangle$.
  - Add the edges $\langle v_j, -^i_{j+1} \rangle$ and $\langle v_j, +^i_{j+1} \rangle$.

  Note that C_0 refers to no clause, and that $-^i_{m+1} = +^i_{m+1} = q^i$.
- Set $i = i + 1$.



(a) Clause-gadget



(b) x_i-gadget

FIGURE 8. The gadgets that make up the initial position of SY-PI(SY_φ). The sub-graph H_j is a fully connected graph with 4 elements, all of whose vertices are connected with the vertex w_j.

The Scotland Yard instance SY_φ is derived from the board game: the digraph is completely spelled out and the initial positions are as indicated in the gadgets. Note that the every vertex in the constructed digraph has at least one outgoing edge. In fact the reflexive edges in s^i and t^i serve merely to accomplish this fact. Therefore, SY_φ is fully specified after putting $f : \{1, \dots, 2m + 2\} \to \{\mathsf{hide}\}$. Hence, SY_φ is an instance of SCOTLAND YARD$_{\mathsf{hide}}$.

It remains to be shown that $\varphi \in$ 3-SAT iff $\text{SY}_\varphi \in$ SCOTLAND YARD$_{\mathsf{hide}}$.

By Theorem 5.7, it is sufficient to show that $\varphi \in$ 3-SAT iff $\exists$ has a winning strategy in SY-PI(SY_φ).

From left to right. Suppose φ is satisfiable, then there exists a truth assignment $t : \{\vec{x}\} \to \{\mathsf{true}, \mathsf{false}\}$ such that for every clause C_j in φ, there exists at least one literal that is true under t. Let us describe a strategy for $\exists$ that is based on t and argue that it is in fact a winning strategy for her in SY-PI(SY_φ):

- If $\exists_i$ stands on the vertex on layer 0 and $t(x_i) = \mathsf{true}$ (false), then move it to $+^i_1$ ($-^i_1$) on layer 1.
- If $\exists_i$ stands on $-^i_j$ ($+^i_j$), and $-^i_j$ ($+^i_j$) happens to be connected to w_j, then move it to w_j. If $\exists_i$ stands on $-^i_j$ ($+^i_j$) and $-^i_j$ ($+^i_j$) is not connected to w_j, then move it to d^i_j (e^i_j).
- If $\exists_i$ stands on w_j, move it to $\pm^i_{j+1}$, for $\pm \in \{+, -\}$. Note that this move is deterministic, since there is an edge from w_j to $+^i_{j+1}$, say, only if x_i occurs as a literal in C_j. By assumption of φ being an instance of 3-SAT, it cannot be the case that also $\neg x_i$ occurs as a literal in C_j. Hence, there is no edge from w_j to $-^i_{j+1}$.
- If $\exists_i$ stands on d^i_j (e^i_j) then move it to $-^i_{j+1}$ ($+^i_{j+1}$). If $\exists_i$ stands on d^i_m or e^i_m then move it to q^i.
- If $\exists_i$ stands on q^i, then move it to s^i if $t(x_i) = \mathsf{true}$ and to t^i if $t(x_i) = \mathsf{false}$.

Observe that if $\exists$ plays according to the above strategy, every pawn $\exists_i$ will eventually traverse either all vertices $-^i_1, \dots, -^i_m$ or all vertices $+^i_1, \dots, +^i_m$, given that $t(x_i) = \mathsf{false}$ or $t(x_i) = \mathsf{true}$, respectively.

To show that this strategy is indeed winning against any of $\forall$'s strategies, consider the sets of vertices U^i_j that $\forall$ occupies on the clause-gadget and the x_i-gadget, after round $0 \leq j \leq 2m+2$ of the game in which $\exists$ moved as described above. Initially, $\forall$ has one pawn on v_0; thus, $U^i_0 = \{v_0\}$. Let us suppose without loss of generality that $t(x_i) = \mathsf{true}$. Then, $U_1^i = \{u_1, -_1^i\}$ as the $\forall$ pawn put on $+_1^i$ is captured by $\exists_i$. We leave it to the reader to check that for $1 \leq j \leq m-1$, it is the case that

$$\begin{aligned} U_{2j}^i &= \{v_j, c_j^i, d_j^i\} \\ U_{2j+1}^i &= \{u_{j+1}, a_{j+1}^i, -_{j+1}^i\}. \end{aligned}$$

The crucial insight being that the $\forall$ pawn put on w_j can be captured iff there exists at least one literal in C_j that is made true by t. Since t was assumed to be a satisfying assignment, there must be at least one $\exists$ pawn that captures the universal pawn on w_j. It is prescribed by the above strategy that $\exists_i$ is moved to any w_j-vertex, if possible. Furthermore, it is required to return to the x_i-gadget on the next round of the game, capturing the $\forall$ pawn that was positioned on $+_{j+1}^i$, from v_j.

After round $2m-1$, $\forall$ cannot continue walking on the *safe path* defined by $v_0, u_1, \ldots, v_m$; indeed, he has one pawn on v_m and two pawns per x_i-gadget: $U_{2m}^i = \{c_m^i, d_m^i\}$. The pawn put on q^i from v_m is captured by $\exists_i$ coming from e_m^i, so we get that $U_{2m+1}^i = \{p^i\}$. Following the strategy above, $\exists$ moves $\exists_i$ from q^i to s^i, whence $U_{2m+2}^i = \varnothing$. Since i was chosen arbitrarily, it is the case that $\forall$ has no pawns left on any x_i-gadget and therefore has lost after exactly $2m+2$ rounds of playing.

From right to left. Suppose φ is not satisfiable, then for every truth assignment t to the variables in φ, there exists a clause C_j in φ, that is made false. In the converse direction of this proof, we concluded that every $\exists_i$ traverses one of the paths $-_1^i, \ldots, -_m^i, q^i$ and $+_1^i, \ldots, +_m^i, q^i$, depending on $t(x_i)$. This behavior We call *in accordance with the truth value $t(x_i)$ assigned to x_i*; if this behavior is displayed with respect to every $1 \leq i \leq n$, then we say that it is *in accordance with the truth assignment t*.

For now, assume that $\exists$ plays in accordance with some truth assignment t. Since φ is not satisfiable, it is not satisfied by t either. Therefore, there is a clause C_j that is not satisfied by t. This is reflected during the playing of the game by the fact that after round $2j$ there is a $\forall$ pawn positioned on w_j that cannot be captured by any $\exists_i$. This state of affairs will result in a win for $\forall$, as he positions pawns on every vertex in H_j during round $2j+1$. By construction, H_j is a connected graph on which he can keep on putting pawns indefinitely.

Remains to be shown that $\exists$ cannot avoid losing by deviating from playing in accordance with some truth assignment. We make the following claims: (A) If after round $1 \leq 2j-1 \leq 2m+1$ there is an i such that no $\exists$ pawn is positioned on $-_j^i$ or $+_j^i$, then $\exists$ loses. (B) If after round $2 \leq 2j \leq 2m$ there is an $\exists$ pawn positioned on c_j^i or f_j^i, then $\exists$ loses. We prove this by induction. While proving these claims, we take the easily derived fact for granted that during round $2j-1$ of the game $\forall$ has a pawn on u_j and that during round $2j$ of the game $\forall$ has a pawn on v_j.

Base step. (A) Suppose after round $2m+1$ no $\exists$ pawn is on q^i (recall that $-^i_{m+1} = +^i_{m+1} = q^i$). Then, there is a $\forall$ pawn on q^i, since by construction of the game board, v_m is connected to q^i. During the next round, $\forall$ has pawns on both s^i and t^i, none of which is captured by $\exists$, as she has no pawns on the x_i-gadget.

(B) Suppose after round $2m$ there is an $\exists$ pawn positioned on c^i_m, say. We make a case distinction regarding the state of affairs after round $2m+1$: (i) there is an $\exists$ pawn on q^i. Obviously, this pawn cannot be the one on c^i_m after round $2m$, since there is no edge from c^i_m to q^i. Therefore there are two of $\exists$'s pawns on the x_i-gadget. As there are exactly n pawns at $\exists$'s disposal, during round $2m+1$ there is an x_h-gadget avoid of $\exists$ pawns. In particular, there is no $\exists$ pawn on q^h. Applying clause (A), yields that $\exists$ cannot win from this position. (ii) There is no $\exists$ pawn on q^i. Then, there is a $\forall$ pawn on q^i after round $2m+1$, coming from v_m. Therefore, after round $2m+2$ there is a $\forall$ pawn on t^i; since $\exists$ can only capture $\forall$'s pawn at s^i.

Induction step. (A) Suppose after round $1 \leq 2j-1 < 2m-1$ there is an i such that no $\exists$ pawn is positioned on $-^i_j$ or $+^i_j$. Since $\forall$ has a pawn on v_{j-1} after round $2j-2$, he has pawns on both $-^i_j$ and $+^i_j$ after round $2j-1$. If after the next round $\forall$ occupies the vertices c^i_j, d^i_j, e^i_j or d^i_j, e^i_j, f^i_j this implies that one of $\exists$'s pawns is on c^i_j or f^i_j, respectively. But then she loses in virtue of the inductive hypothesis of (B). So, suppose that after round $2j$ $\forall$ occupies all the vertices $c^i_j, d^i_j, e^i_j, f^i_j$. Then, for the x_i-gadget to be cleansed of $\forall$ pawns it is prescribed that on some later round of the game there are at least two $\exists$ pawns on this gadget. But then on this round the inductive hypothesis of (A) applies, yielding that $\exists$ loses.

(B) Suppose after round $2 \leq 2j < 2m-2$ there is an $\exists$ pawn positioned on c^i_j, say. Then, after round $2j+2$ the same pawn is positioned on c^i_{j+1}. Applying the inductive hypothesis of (B) teaches that $\exists$ loses.

We leave it to the reader to check that if $\exists$ plays in such a way that if during any of the rounds of the game the premises of (A) and (B) do not apply, then she plays in accordance with some truth assignment. However, also playing according to any truth assignment is bound to be a losing way of playing, as we argued earlier. This concludes the proof. Q.E.D.

Tying together the latter two theorems yields **NP**-completeness for the specialization of Scotland Yard in which $\forall$ does not give any information.

Theorem 7.3. Scotland Yard$_{\mathsf{hide}}$ is **NP**-complete.

Proof. Immediate from Lemmas 7.1 and 7.2. Q.E.D.

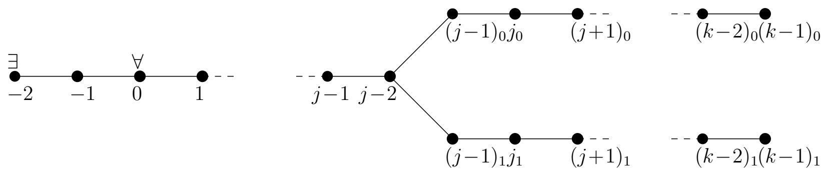


FIGURE 9. The forked graph F. $\exists$ has a winning strategy iff she knows $\forall$'s position during round j.

From a computational point of view it is easier to solve the decision problem SCOTLAND YARD, when $\forall$ does not reveal himself during the game. Yet, in a quantitative sense it becomes harder for $\exists$ to play this game, in that there are games in which $\exists$ has no winning strategy if $\forall$ does not reveal himself at all, but she would have had a winning strategy if $\forall$ was to reveal himself at least once. To make this claim precise fix two functions g and h, where

$$g : \{1, \ldots, k\} \to \{\mathsf{hide}\} \quad \text{and} \quad h : \{1, \ldots, k\} \to \{\mathsf{hide}, \mathsf{show}\}$$

such that $h(j) = \mathsf{show}$, for some j. We leave it to the reader to check that $\exists$ has a winning strategy in $\mathrm{SY}(F, \langle u_*, \vec{v}_* \rangle, h)$ but none in $\mathrm{SY}(G, \langle u_*, \vec{v}_* \rangle, g)$. In the latter games, F is the graph depicted in Figure 9.

8 Concluding remarks

By means of a power set construction we observed that imperfect information of vertices can be propagated to perfect information of sets of vertices, without affecting the property of the cops having a winning strategy, cf. Theorem 5.7. Lemma 6.1 shows that the decision problem SCOTLAND YARD$_{\mathsf{show}}$ is **PSPACE**-hard; Theorem 6.3 shows that that it is **PSPACE**-complete. This finding is in line with the literature on combinatorial game theory, since the former decision problem concerns two-player graph games with perfect information. More surprisingly, it was shown in Lemma 6.1 that the power set analysis does not come at a computational cost: also SCOTLAND YARD is solvable in **PSPACE**.

The question why, on an abstract level, the imperfect information in Scotland Yard does not increase the computational complexity is not addressed in this paper. Thus, a direction for future research is to explore what are Scotland Yard's properties that cause it to behave like most two-player games with perfect information.

We made the point that under the current analysis, Scotland Yard games enjoy the same level of abstraction as graph games. Still the games under consideration form a coherent lot. To name some of their shared properties: the duration is bound by the graph's size; the imperfect information satisfies perfect recall; the information function cannot account for very subtle patterns of ignorance; and, the graphs were only supposed to have out-degree ≥ 1. Thus a more general theory is desirable that charts the computational landscape of imperfect information graph games. In particular the question is worthwhile under what conditions the complexity of graph games does not increase when imperfect information is inserted.

References

[Ah_0+73] A.V. Aho, A.B. Borodin, R.L. Constable, R.W. Floyd, M.A. Harrison, R.M. Karp & H.R. Strong, eds. *Proceedings of the fifth annual ACM symposium on Theory of computing 1973, Austin, Texas, United States, April 30–May 02, 1973.* ACM Press, 1973.

[$Be_1Co_3Gu_2$82] E.R. Berlekamp, J.H. Conway & R.K. Guy. *Winning Ways.* Academic Press, 1982.

[Bi_0+02] T.C. Biedl, E.D. Demaine, M.L. Demaine, R. Fleischer, L. Jacobsen & J.I. Munro. The complexity of Clickomania. In [$No_1$02, pp. 389–404].

[$Ca_1Ho_1$01] P.J. Cameron & W. Hodges. Some combinatorics of imperfect information. *Journal of Symbolic Logic* 66(2):673–684, 2001.

[$Ch_0Ko_6St_6$81] A.K. Chandra, D.C. Kozen & L.J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery* 28(1):114–133, 1981.

[$Fr_0$02] A.S. Fraenkel. Combinatorial games: selected bibliography with a succinct gourmet introduction. In [$No_1$02, pp. 475–535].

[Fr_0+78] A.S. Fraenkel, M.R. Garey, D.S. Johnson, T. Schäfer & Y. Yesha. The complexity of checkers on an $N \times N$ board – preliminary report. In [Ly78, pp. 55–64].

[$Fr_0Li_1$81] A.S. Fraenkel & D. Lichtenstein. Computing a perfect strategy for $n \times n$ chess requires time exponential in n. *Journal of Combinatorial Theory, Series A* 31(2):199–214, 1981.

[Go$_1$Re$_2$95] A.S. Goldstein & E.M. Reingold. The complexity of pursuit on a graph. *Theoretical Computer Science* 143(1):93–112, 1995.

[Ho$_0$Lo$_3$13] E.W. Hobson & A.E.H. Love, eds. *Proceedings of the Fifth International Congress of Mathematicians, Vol. 2.* Cambridge University Press, 1913.

[Ho$_1$97a] W. Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL* 5(4):539–563, 1997.

[Ho$_4$Ul79] J.E. Hopcroft & J.D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[Ka$_5$00] R. Kaye. Minesweeper is NP-complete. *Mathematical Intelligencer* 22(2):9–15, 2000.

[Ko$_3$Me$_0$92] D. Koller & N. Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior* 4(4):528–552, 1992.

[Li$_1$Si$_2$80] D. Lichtenstein & M. Sipser. GO is polynomial-space hard. *Journal of the Association for Computing Machinery* 27(2):393–401, 1980.

[Ly78] N.A. Lynch, ed. *Proceedings of the 19th Annual Symposium on the Foundations of Computer Science.* IEEE Computer Society, 1978.

[No$_1$02] R.J. Nowakowski, ed. *More Games of No Chance*, *MSRI Publications* 42. Cambridge University Press, 2002.

[Pa$_4$94] C.H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[Pe$_3$AzRe$_1$01] G. Peterson, S. Azhar & J.H. Reif. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers and Mathematics with Applications* 41(7–8):957–992, 2001.

[Re$_1$84] J.H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences* 29(2):274–301, 1984.

[$\text{Sc}_0$78] T.J. Schäfer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences* 16(2):185–225, 1978.

[$\text{Se}_2$06] M. Sevenster. *Branches of imperfect information: logic, games, and computation.* Ph.D. thesis, Universiteit van Amsterdam, 2006. *ILLC Publications* DS-2006-06.

[$\text{St}_6\text{Me}_2$73] L.J. Stockmeyer & A.R. Meyer. Word problems requiring exponential time. In [Ah_0+73, pp. 1–9].

[$\text{vNMo}_0$44] J. von Neumann & O. Morgenstern. *Theory of games and economic behavior.* John Wiley and Sons, 1944.

[Ze13] E. Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachpiel. In [$\text{Ho}_0\text{Lo}_3$13, pp. 501–504].

Approaches to Independence Friendly Modal Logic*

Tero Tulenheimo[1]
Merlijn Sevenster[2]

[1] Filosofian laitos
Helsingin yliopisto
PL 9
00014 Helsinki, Finland

[2] Philips Research
Prof. Holstlaan 4
5656 AA Eindhoven, The Netherlands

`tero.tulenheimo@helsinki.fi, merlijn.sevenster@philips.com`

Abstract

The aim of the present paper is to discuss two different approaches for formulating independence friendly (IF) modal logic. In one of them, the language of basic modal logic is enriched with the slash notation familiar from IF first-order logics, and the resulting logic is interpreted in terms of games and uniform strategies. A different approach is formulated in the present paper: an IF modal logic is defined by imposing conditions on its structural relationships to other logics, namely a specified modal logic (say, basic modal logic), its first-order correspondence language, and IF logic. We compare logics emerging from the two approaches. More generally, the issue of the *Eigenart* of IF modal logics is addressed.

1 Introduction

Already in the seminal publications on *independence friendly first-order logic* (IF logic) [$\text{Hi}_1$95, $\text{Hi}_1$96, $\text{Hi}_1\text{Sa}_4$89, $\text{Sa}_4$93], applications were pointed out involving a first-order modal setting. It was argued that the logical form of some natural language sentences is best captured by formulas that allow for *slashing* relative to modal operators—marking certain logical operators as independent of modal operators in whose syntactic scope they nevertheless lie. In the first publications that developed an independence friendly modal logic, Bradfield [$\text{Br}_0$00] together with Fröschle [$\text{Br}_0\text{Fr}_4$02a]

* We wish to thank Peter van Emde Boas, Tapani Hyttinen and Jouko Väänänen for helpful discussions, and the anonymous referee for most useful suggestions and criticisms.

interpreted the logic's independence indications using a combination of transition systems with *concurrency* and games of *imperfect information.* Tulenheimo [$Tu_1$03, $Tu_1$04] together with Hyttinen [$Hy_1Tu_1$05] showed that a reasonable IF modal logic can be defined simply using Hintikka's original idea of implementing logical independence by informational independence in the sense of game theory [$Hi_1$73, $Hi_1$95, $Hi_1$96]. To model logical independence, this suffices; it is not necessary to enrich standard modal structures by introducing concurrency as a separate, primitive component of the models. This type of study of IF modal logic serves to attract interest in the larger program of independence-friendliness that investigates the notion of informational independence in logic.

The aims of the present paper are twofold. (1) First, we wish to give a recap of the various logics introduced under the heading 'IF modal logic' whose semantics rely simply on the game-theoretical notion of informational independence, as just explained. (Accordingly, we do not discuss Bradfield's independence friendly modal logics.) The pre-theoretical motivation for all these logics was, when they were introduced, that they would be 'modal analogues' of IF first-order logic—in syntax as well as in semantics.

The logics termed 'independence friendly modal logic' in the relevant research publications [$Hy_1Tu_1$05, $Tu_1$03, $Tu_1$04, $Tu_1Se_2$06] differ among themselves both in syntax and in semantics. (The sense of diversity is of course only increased when the logics of [$Br_0$00, $Br_0Fr_4$02a] are considered as well.) Depending on which syntactic restrictions one imposes on the formation of the independence indications, logics with different metalogical properties result. As will turn out, the appropriate properties may diverge strikingly from case to case (cf. Table 2 in Section 6). The present authors conclude that a framework is desirable in which the various systems can be systematically studied and compared. In this vein, our second aim (2) is to introduce a framework that sheds a unifying light on the IF modal logics introduced so far, and can be used to develop new logics that are both modal and independence friendly. The framework we put forward is determined by three parameters. These parameters are inspired by the standard translation of basic modal logic into first-order logic, and Hintikka's IF procedure that brings us from first-order logic to IF first-order logic.

Although we find our framework a natural environment for studying independence friendliness and modal logic, by no means do we claim that the framework covers all conceivable IF modal logics. Neither do we claim that all logics to be found within this framework are equally interesting. In fact, we regard it as one of the virtues of the framework that within its confines, one can isolate logics some of which enjoy 'nicer' properties than others. From this very perspective, we define the so-called 'structurally determined IF modal logic'. In [$Tu_1Se_2$06] this logic is shown to combine a number of nice properties: strong expressive power, decidability, and allowing for a compositional semantics.

Let us now introduce some basic notions, and fix the plan of the paper.

IF first-order logic. It has been observed by Hintikka, on various occasions, that as a matter of fact, the *syntactic scope* and *(logical) priority scope* of quantifiers coincide in first-order logic. Hintikka [Hi$_1$96] points out that there is no general pre-theoretical backing for this assumption, and provocatively refers to it as *Frege's fallacy.* The formulas of IF first-order logic, denoted **IF**, carry the slash '/' as a new item of notation. The slash, as in $\forall y(\exists x/y)\varphi$, is to be interpreted in such a way that the occurrence of $\exists x$ is outside the logical priority scope of $\forall y$, although it falls within the syntactic scope of $\forall y$. The formulas of **IF** are generated from the fragment of first-order logic in which every variable is quantified at most once and in which every formula is in *negation normal form*, to be denoted **FO**. Formally, we let **IF** be the smallest superset of **FO** closed under the following condition:

- If $\varphi \in$ **IF** and $\exists x$ occurs in φ in the syntactic scope of quantifiers among which $Q_1 y_1, \ldots, Q_n y_n$, then the formula resulting from replacing $\exists x$ by $(\exists x/y_1, \ldots, y_n)$ is also in **IF**,

where $Q_i y_i$ stands for $\forall y_i$ or $\exists y_i$. The notion of 'binding a variable' is extended from the usual first-order case by saying that the quantifier $Q_i y_i$ *binds* the occurrence of the variable y_i in $(\exists x/y_1, \ldots, y_n)$, with $1 \leq i \leq n$. We write $\exists x$ rather than $(\exists x/\varnothing)$, if the tuple $y_1, \ldots, y_n$ is empty. One may consider the above rule as specifying an *IF procedure*, producing **IF** from **FO**. In the literature various other IF procedures are put forward, that allow for marking propositional connectives as independent of quantifiers, marking quantifiers as independent of (suitably construed) propositional connectives, and/or marking universal quantifiers as independent of other logical operators. In the current paper, we refrain from considering these options.

Basic modal logic. Formulas of *basic modal logic* (**ML**) are generated from a fixed class **prop** of propositional atoms by the following grammar:

$$\varphi ::= p \mid \neg p \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid \Diamond\varphi \mid \Box\varphi,$$

where $p \in$ **prop**. Its semantics is defined relative to *modal structures* and their states, that is, tuples $\mathfrak{M} = (M, R, V)$ and elements $w \in M$, where M is a non-empty domain, R is a binary relation on M termed *accessibility relation*, and V : **prop** $\longrightarrow Pow(M)$ is a *valuation function.* It will be assumed that the clauses recursively associating a truth condition with all **ML**-formulas are familiar. (The reader may consult, e.g., [BldRVe$_1$01, Section 1.3].) *Polymodal basic modal logic* $\mathbf{ML}_k$ is like $\mathbf{ML}$, but involves k modality types, each with its own box $\Box_i$ and diamond $\Diamond_i$. Its semantics is in terms of k-ary modal structures $(M, R_1, \ldots, R_k, V)$, having for every modality type i an accessibility relation $R_i \subseteq M^2$ of its own.

Expressive power. If L and L' are two modal logics whose semantics are defined relative to a class $\mathcal{K}$ of modal structures, we say that L is *translatable into* L' over $\mathcal{K}$ (in symbols $L \leq_{\mathcal{K}} L'$), if for every $\varphi \in L$, there is $\psi_\varphi \in L'$ such that for all modal structures $\mathfrak{M} \in \mathcal{K}$ and all $w \in M$, we have: $\mathfrak{M}, w \models \varphi$ if, and only if, $\mathfrak{M}, w \models \psi_\varphi$. L' is *more expressive than* L over $\mathcal{K}$, or has *greater expressive power than* L over $\mathcal{K}$ (symbolically $L <_{\mathcal{K}} L'$), if $L \leq_{\mathcal{K}} L'$ but $L' \not\leq_{\mathcal{K}} L$. The logics L and L' have the *same expressive power* over $\mathcal{K}$, or *coincide* over $\mathcal{K}$ (denoted $L =_{\mathcal{K}} L'$), if $L \leq_{\mathcal{K}} L'$ and $L' \leq_{\mathcal{K}} L$. When speaking of the class of *all* modal structures, we suppress the subscript indicating the class altogether, and write simply $L \leq L'$ and so on.

These notions are naturally extended to a comparison between a modal logic and (IF) first-order logic. To every modal structure $\mathfrak{M} = (M, R, V)$ there corresponds, in a canonical way, a first-order structure $\mathfrak{M}^{\mathbf{FO}} = (M, R, \langle V(p)\rangle_{p \in \mathbf{prop}})$, interpreting a binary relation symbol R as the binary relation R, and, for each $p \in \mathbf{prop}$, a unary relation symbol P as the set $V(p)$. Saying, for instance, that L is translatable into $\mathbf{FO}$, means that for every $\varphi \in L$ there is a first-order formula $\psi_\varphi(x)$ of one free variable, x, written in the vocabulary $\{\mathrm{R}, \langle \mathrm{P}\rangle_{p \in \mathbf{prop}}\}$, such that for all modal structures $\mathfrak{M}$ and all $w \in M$, we have: $\mathfrak{M}, w \models \varphi$ iff $\mathfrak{M}^{\mathbf{FO}}, \gamma \models \psi_\varphi$, where $\gamma(x) = w$.

Plan of the paper. In Sections 2 and 3 we survey two IF modal logics interpreting the slash device in terms of informational independence, referred to as **IFML** and $\mathbf{EIFML}_k$. As an original result we prove Theorem 2.6, stating that **IFML** cannot be translated into first-order logic. The logics **IFML** and $\mathbf{EIFML}_k$ show that allowing independence friendliness serves to increase the expressive power of a modal logic. However, the definitions of these logics also suggest that many more IF modal logics can be obtained by varying the syntax and the IF procedure applied.

In Section 4 we propose a new framework for studying IF modal logics from the IF first-order viewpoint. Essentially, the framework allows for systematically varying the syntax and the IF procedure used in defining an IF modal logic. We discuss at some length a particular logic obtained in this framework, termed 'structurally determined IF modal logic', or $\mathbf{IFML}_{\mathbf{SD}}$. (This logic is extensively studied by the authors in [Tu$_1$Se$_2$06].) To give a fuller picture of the expressivity of the various IF modal logics discussed in the paper, in Section 5 we provide an original negative expressivity result concerning **IFML**, $\mathbf{EIFML}_k$ and $\mathbf{IFML}_{\mathbf{SD}}$, proving that relative to a certain class of trees, the expressive power of all these logics collapses to that of basic modal logic. Section 6 serves as a conclusion in which we comment the issue of informational independence in logics, putting forward our conviction that the notion of informational independence not only makes sense with respect to logics other than first-order logic (since it can, for instance, be systematically studied in connection with modal logic) but also enjoys general theoretical interest. In this concluding section also a summary of known results concerning different IF modal logics can be found, as well as a table where some conjectures about them are listed.

Note on notation. If L is a logic for which syntax and semantics is defined, and φ is a formula of L, we write $\varphi \in L$ to say that φ is among the formulas of L. That is, when no confusion may arise, we do not notationally distinguish a logic from its set of formulas. By "L-formula" we mean formula of L.

2 IF modal logic via independence indications

We wish to introduce a modification of basic modal logic where diamonds may be 'indicated as independent' from any syntactically superordinate modal operators (boxes or diamonds). Such indicating is accomplished by using a notation $(\Diamond/i_1, \ldots, i_k)$, where $i_1, \ldots, i_k$ are positive integers which in a specified, systematic way identify superordinate modal operators. Such syntactic independence indications are semantically interpreted in terms of 'logical dependence': the choice of a state as a semantic value of a diamond $(\Diamond/i_1, \ldots, i_k)$ must not depend on the states interpreting the modal operators identified by the integers $i_1, \ldots, i_k$. Supposing that $(a_1, \ldots, a_n)$ and $(a'_1, \ldots, a'_n)$ are two sequences of choices for modal operators superordinate to $(\Diamond/i_1, \ldots, i_k)$, then if these sequences agree on all choices save for those corresponding to the operators identified by the integers $i_1, \ldots, i_k$, the state chosen for $(\Diamond/i_1, \ldots, i_k)$ must be the same in both cases.

The logic we now proceed to define is dubbed *independence friendly modal logic*. We stress that it carries this name for 'historical reasons'—by no means do we wish to suggest that this logic is *the* IF modal logic. Semantically, IF modal logic will emerge as a proper extension of basic modal logic. This observation increases interest in the study of IF modal logics, for it gives rise to the hope that independence friendliness is a dimension of modal logics that may yield more expressive, yet decidable systems. (That entertaining such a hope is not entirely unrealistic can be seen from the decidability results concerning certain specific IF modal logics, cf. [Hy$_1$Tu$_1$05, Tu$_1$Se$_2$06].) We now turn to defining the syntax and semantics of this logic in detail.

2.1 Definition of the logic

Syntax. The formulas of *Independence friendly (IF) modal logic* (**IFML**) are obtained from those of **ML** by the following rewriting rules:

1. If $\psi \in \mathbf{ML}$, then the result of replacing all occurrences of $\Diamond$ in ψ by the symbol $(\Diamond/\varnothing)$ is a formula.
2. If ψ is a formula, $(\Diamond/\varnothing)$ appears in ψ, and $i_1, \dots, i_n$ is a tuple of positive integers, then the result of replacing that token of $(\Diamond/\varnothing)$ in ψ by the symbol $(\Diamond/i_1, \dots, i_n)$ is also a formula.

Formulas of **IFML** are precisely the strings generated by the above two rules. By stipulation, we write $\Diamond$ for $(\Diamond/\varnothing)$. Thereby any string that is a formula of **ML**, is in fact also a formula of **IFML**. Note that the input and output of the above rule 2 are identical in the special case that $n := 0$. Examples of **IFML**-formulas are:

$$\begin{array}{lll} \Box\Diamond p, & \Box(\Diamond/1)p, & \Diamond(\Diamond/1)p, \\ \Box(\Diamond/127)\Box(\Diamond/1,2)p, & (\Box(\Diamond/1)p \wedge \Diamond\Box(\Diamond/1,2)q), & \Box(p \vee (\Diamond/1)q), \\ \Box(p \wedge (\Diamond/1)q). & & \end{array}$$

It was already pointed out that the role of the integers $i_1, \dots, i_n$ following a diamond sign, as in $(\Diamond/i_1, \dots, i_n)$, is to *identify* certain syntactically superordinate modal operators. Which ones? The principle of identification we make use of, is based on the left-linear relation of *syntactic subordination* among tokens of operators $(\vee, \wedge, \Diamond, \Box)$ appearing in formulas $\varphi \in \mathbf{IFML}$. Relative to a formula φ, this relation induces a tree structure, with the unique outmost operator of φ at its root, and operators with no subordinate operators at leaves. Hence for any operator-token, the set of its predecessors in this tree structure determines a *linear* order. If $O \in \{\vee, \wedge, \Diamond, \Box\}$ appears in φ, it is either itself the unique outmost operator of φ, or else there is a unique immediate predecessor O' of O among the operator-tokens to which O is subordinate, and so on. So we may speak of 'the n-th predecessor' of O. We can also restrict attention to *modal* operators preceding O, and enumerate them beginning from the one that is furthest and ending up with the one that is closest. In this way we may speak of 'the n-th modal operator in φ among those modal operators that precede O'—hence counting only modal operators and ignoring conjunctions and disjunctions. It is to the numbers identifying the locations of modal operators syntactically superordinate to $(\Diamond/i_1, \dots, i_n)$ in this latter type of numbering, that the integers $i_1, \dots, i_n$ refer.

In $\Box(\Diamond/1)\Box(\Diamond/1,2)p$, the numeral 1 in $(\Diamond/1)$ refers to the immediately preceding box, and the numerals 1 and 2 in $(\Diamond/1,2)$ to the first occurrence of $\Box$ *resp.* to the first (and only) occurrence of $(\Diamond/1)$. In $(\Box(\Diamond/1)p \wedge \Diamond\Box(\Diamond/1,2)q)$, the numeral 1 in $(\Diamond/1)$ identifies the box in the left conjunct, whereas the same numeral identifies the outmost diamond of the right conjunct in $(\Diamond/1,2)$. We allow for vacuous identifiers: in

$\Box(\Diamond/127)p$ the numeral 127 refers to nothing at all, since there are no 126 or more nested modal operators syntactically preceding $\Box$ in that formula.

In earlier publications on IF modal logic [Br$_0$00, Br$_0$Fr$_4$02a, Hy$_1$Tu$_1$05, Tu$_1$03, Tu$_1$04], various different identification methods are used for singling out the desired superordinate modal operators. Typically this has been accomplished by introducing an explicit indexing or labelling of tokens of modal operators as a part of the syntax. The possibility of defining the syntax as above shows that such an indexing is not a conceptually necessary ingredient of IF modal logics.[1]

The set Sub$[\varphi]$ of *subformulas* of a formula $\varphi \in$ **IFML** is defined in a straightforward way: Sub$[p] = \{p\}$ and Sub$[\neg p] = \{\neg p\}$; for $\circ \in \{\vee, \wedge\}$: Sub$[(\psi \circ \theta)] = \{(\psi \circ \theta)\} \cup$ Sub$[\psi] \cup$ Sub$[\theta]$; Sub$[\Box\psi] = \{\Box\psi\} \cup$ Sub$[\psi]$; and Sub$[(\Diamond/i_1, \ldots, i_n)\psi] = \{(\Diamond/i_1, \ldots, i_n)\psi\} \cup$ Sub$[\psi]$. A formula $\varphi \in$ **IFML** is *closed*, if it contains no vacuous identifiers, i.e., if every $(\Diamond/i_1, \ldots, i_n)$ appearing in φ is subordinate to at least $\max\{i_1, \ldots, i_n\}$ nested modal operators in φ. Otherwise φ is *open*.

Semantics. There may appear in a given formula many *tokens* of the same subformula. (E.g., in $(p \vee p)$ there appear two tokens of the subformula p.) When defining the semantics of an IF logic, one must pay specific attention to this fact, to be able to formulate clauses defining the semantic role of operators with independencies, such as $(\Diamond/i_1, \ldots, i_k)$.

We follow [Vä07] in understanding formulas explicitly as finite *strings of symbols*. Each numeral standing for a positive integer in a formula of **IFML** is counted as a separate symbol (no matter how many digits it has in the chosen presentation), other symbols being propositional atoms,), (, $\neg$, $\vee$, $\wedge$, $\Diamond$, $\Box$, $\varnothing$, the comma and the slash-sign /. The *length* of a string S, in symbols $|S|$, is the number of symbols in S when each symbol is counted as many times as it occurs. The symbols appearing in a formula are *enumerated* with positive integers starting from left to right.

For illustration, consider the formula $\varphi := \Box(\Diamond/1, 27)(p \vee q)$.

$\Box$	(	$\Diamond$	/	1	,	27	)	(	p	$\vee$	q	)
1	2	3	4	5	6	7	8	9	10	11	12	13

In the special case that the nth symbol of a string ψ starts itself a string which is a subformula of ψ, we write $\Lambda(\psi, n)$ for that subformula. In the above example, $\Lambda(\varphi, 9) = (p \vee q)$ and $\Lambda(\varphi, 10) = p$. Every subformula of a formula ψ is of the form $\Lambda(\psi, n)$ for some n, and some subformulas may appear in ψ corresponding to several numbers n. It may further be noted that if ψ is closed and the operator $(\Diamond/i_1, \ldots, i_n)$ appears in ψ, the above

[1] Essentially this definition of the syntax suggested to one of the authors (TT) by Balder ten Cate in December 2002.

enumeration of the symbols in ψ could be used as an alternative way of identifying those modal operators, superordinate to $(\Diamond/i_1, \ldots, i_n)$, that by syntax are identified by the integers $i_1, \ldots, i_n$.

In defining game-theoretical semantics for **IFML**, we adapt the definition that is given in [Vä07] for IF first-order logic. For every formula φ, modal structure $\mathfrak{M}$ and state $w_0 \in M$, a semantic game $G(\varphi, \mathfrak{M}, w_0)$ between two players ($\exists$ and $\forall$) is associated by defining the set of its *positions*. We refer to $\exists$ as 'she' and to $\forall$ as 'he'. If $\varsigma = (a_0, \ldots, a_n)$ is a finite sequence, we write $\max(\varsigma)$ for its last member, $\max(\varsigma) := a_n$. If a_{n+1} is any further object, we write $\varsigma^\frown a_{n+1}$ for the extended sequence $(a_1, \ldots, a_n, a_{n+1})$.

Definition 2.1 (Positions). *Positions* are triples (ψ, n, ς), where $\psi = \Lambda(\varphi, n)$ and ς is a finite sequence of elements of M. In the beginning of the game the position is $(\varphi, 1, \langle w_0 \rangle)$. The following conditions serve to generate the set of all positions of $G(\varphi, \mathfrak{M}, w_0)$, with $\mathfrak{M} = (M, R, V)$. They also specify which player makes which type of choice (if any) at a given position.

1. (a) If (p, n, ς) is a position, then: if $\max(\varsigma) \in V(p)$, $\exists$ wins, otherwise $\forall$ wins. (b) If $(\neg p, n, \varsigma)$ is a position, then: if $\max(\varsigma) \notin V(p)$, $\exists$ wins, else $\forall$ wins.

2. If $((\psi \vee \varphi), n, \varsigma)$ is a position, also $(\psi, n+1, \varsigma)$ and $(\varphi, n+2+|\psi|, \varsigma)$ are positions. Player $\exists$ chooses one of these positions at $((\psi \vee \varphi), n, \varsigma)$.

3. If $((\psi \wedge \varphi), n, \varsigma)$ is a position, also $(\psi, n+1, \varsigma)$ and $(\varphi, n+2+|\psi|, \varsigma)$ are positions. Player $\forall$ chooses one of these positions at $((\psi \wedge \varphi), n, \varsigma)$.

4. If $((\Diamond/i_1, \ldots, i_k)\varphi, n, \varsigma)$ is a position and $\langle \max(\varsigma), v \rangle \in R$, then

$$(\varphi, n + 2k + 3 + \sharp(k), \varsigma^\frown v)$$

   is a position, where $\sharp(k) = 0$, if $k \geq 1$ and $\sharp(k) = 2$, if $k = 0$.[2] If there is at least one such position, player $\exists$ chooses one among them at $((\Diamond/i_1, \ldots, i_k)\varphi, n, \varsigma)$. If there is none, player $\forall$ wins.

5. If $(\Box\varphi, n, \varsigma)$ is a position and $\langle \max(\varsigma), v \rangle \in R$, then $(\varphi, n+1, \varsigma^\frown v)$ is a position. If there is at least one such position, player $\forall$ chooses one among them at $(\Box\varphi, n, \varsigma)$. If there is none, player $\exists$ wins.

[2] If $k \geq 1$, the number $n + 2k + 3 + \sharp(k) = n + 2k + 3$ is obtained by counting two parentheses, the diamond, the slash, and k numerals together with $k-1$ commas in the independence indication. However, if $k = 0$, then $(\Diamond/i_1, \ldots, i_k) = (\Diamond/\varnothing)$, and the correct identifier is $n + 2k + 3 + 2 = n + 5$.

Note that the subformula component ψ in a position (ψ, n, ς) is strictly speaking superfluous, because this subformula is fully determined by the number n: $\psi = \Lambda(\varphi, n)$. It is written down here for clarity of exposition.

The above definition of the set of positions in fact serves to define the game $G(\varphi, \mathfrak{M}, w_0)$. This game is a determined zero-sum game of perfect information. We are not, however, interested in who has a winning strategy in this game. What interests us, instead, is who has a strategy that leads to a win against any sequence of moves by the opponent, *and* satisfies the extra condition of *uniformity*, to be defined shortly. The uniformity requirement will have the consequence that to force the desired outcome, player $\exists$, in particular, must make her choices in a 'universalizable' manner: make the same choice in several 'equivalent' circumstances.[3]

Before we can define the uniformity requirement, let us define the notion of game tree; tell what the strategies of the players are; and specify what it means for a player to use a strategy.

Definition 2.2 (Game tree, play, partial play)**.** The set of positions of a semantic game $G(\varphi, \mathfrak{M}, w_0)$ determines, in a canonical way, a tree—to be called the *game tree*. The nodes of the tree are the positions, and its ordering relation is the transitive closure of the relation 'is a successor position of', itself in effect given by the definition of position when telling which are the positions to which a given position gives rise. Any (maximal) branch of the tree represents a possible *play* of the game. Initial segments of plays are called *partial plays*. Sometimes partial plays will be termed *histories* of the game.

Definition 2.3 (Strategy, using a strategy)**.** A *strategy* of player $\exists$ in semantic game $G(\varphi, \mathfrak{M}, w_0)$ is any finite sequence σ of functions σ_i (called *strategy functions*), defined on the set of all partial plays $(p_0, \ldots, p_{i-1})$ satisfying:

- If $p_{i-1} = ((\psi \vee \varphi), n, \varsigma)$, then σ tells $\exists$ which formula to pick, that is, $\sigma_i(p_0, \ldots, p_{i-1}) \in \{n+1, n+2+|\psi|\}$. If the strategy gives the lower value, player $\exists$ picks the left-hand formula ψ, otherwise the right-hand formula φ.

- If $p_{i-1} = ((\Diamond/i_1, \ldots, i_k)\varphi, n, \varsigma)$, then σ tells $\exists$, if possible, which element $v \in M$ with $\langle \max(\varsigma), v\rangle \in R$ to pick. Hence $\sigma_i(p_0, \ldots, p_{i-1}) \in M$ and is accessible from $\max(\varsigma)$. If no suitable element exists, the play has come to an end, with $\forall$ winning the play. So in that case $(p_0, \ldots, p_{i-1})$ is a play, and σ_i is not defined on it.

[3] The resulting game resembles in many respects games of imperfect information, but strictly speaking is not one. This feature of the semantic games for IF modal logic is discussed in [Tu$_1$04, Section 2.3.1].

It is said that $\exists$ *has used strategy* σ in a play of $G(\varphi, \mathfrak{M}, w_0)$, if in each relevant case $\exists$ has made her choice using σ. More exactly, player $\exists$ has used σ in a play $(p_0, \ldots, p_n)$, if the following two conditions hold for all $i < n$:

1. If $p_{i-1} = ((\psi \vee \varphi), m, \varsigma)$ and $\sigma_i(p_0, \ldots, p_{i-1}) = m + 1$, then $p_i = (\psi, m + 1, \varsigma)$, whereas if $\sigma_i(p_0, \ldots, p_{i-1}) = m + 2 + |\psi|$, then $p_i = (\varphi, m + 2 + |\psi|, \varsigma)$.

2. If $p_{i-1} = ((\Diamond/i_1, \ldots, i_k)\varphi, m, \varsigma)$ and $\sigma_i(p_0, \ldots, p_{i-1}) = v$, then $p_i = (\varphi, m + 2k + 3 + \natural(k), \varsigma^\frown v)$.

The notions of strategy and using a strategy can be analogously defined for player $\forall$.

Definition 2.4 (Uniform strategy, winning strategy). A strategy σ of player $\exists$ in semantic game $G(\varphi, \mathfrak{M}, w_0)$ is *uniform*, if the following condition holds. Suppose $p_{i-1} = (\Lambda(\psi, m), m, \varsigma)$ and $p'_{i-1} = (\Lambda(\psi, m), m, \varsigma')$ are two positions arising in the game, when $\exists$ has played according to σ. Further, assume that

$$\Lambda(\psi, m) = (\Diamond/i_1, \ldots, i_k)\varphi.$$

Then if the sequences ς and ς' agree on their values for all arguments except possibly on $i_1, \ldots, i_k$, the strategy σ agrees on the positions $(\Lambda(\psi, m), m, \varsigma)$ and $(\Lambda(\psi, m), m, \varsigma')$, that is to say, $\sigma_i(p_0, \ldots, p_{i-1}) = \sigma_j(p'_0, \ldots, p'_{i-1})$.

A strategy σ of player $\exists$ in game $G(\varphi, \mathfrak{M}, w_0)$ is a *winning strategy*, if σ is uniform, and player $\exists$ wins every play in which she has used the strategy.

The analogous uniformity condition for strategies of $\forall$ is vacuous, since by the syntax, there are no operators of the form $(\Box/i_1, \ldots, i_n)$. A strategy σ of player $\forall$ is winning simply if it leads to a win by $\forall$ against every sequence of moves by $\exists$.

On the basis of the definition of position, the sequences ς and ς' mentioned in the definition of uniformity indeed necessarily have the same length. That is, there is an initial segment Σ of ω such that ς and ς' both are functions of type $\Sigma \longrightarrow M$. If some or all of the numbers $i_1, \ldots, i_k$ happen to be outside of the domain Σ, the sequences ς and ς' are vacuously uniform in the corresponding arguments.

Truth and falsity of **IFML**-formulas are defined as follows:

- φ is true in $\mathfrak{M}$ at w (denoted $\mathfrak{M}, w \models^+ \varphi$), if there is a winning strategy for $\exists$ in $G(\varphi, \mathfrak{M}, w)$.

- φ is false in $\mathfrak{M}$ at w (denoted $\mathfrak{M}, w \models^- \varphi$), if there is a winning strategy for $\forall$ in $G(\varphi, \mathfrak{M}, w)$.

$$\begin{tikzcd}
c_1 & & c_2 & & c_3 & & c_1' & & c_2' & & c_3' \\
& b_1 \arrow[ul, "p"] \arrow[ur] & & b_2 \arrow[ul] \arrow[ur, "p"'] & & & & b_1' \arrow[ul] \arrow[ur, "p"'] & & b_2' \arrow[ul] \arrow[ur] & \\
\mathfrak{M} & & a \arrow[ul] \arrow[ur] & & & & & & a' \arrow[ul] \arrow[ur] & & \mathfrak{N}
\end{tikzcd}$$

FIGURE 1.

- φ is non-determined in $\mathfrak{M}$ at w (denoted $\mathfrak{M}, w \models^0 \varphi$), if game $G(\varphi, \mathfrak{M}, w)$ is not determined, i.e., if neither $\mathfrak{M}, w \models^+ \varphi$ nor $\mathfrak{M}, w \models^- \varphi$.

In what follows we will almost exclusively be interested in *truth* of modal formulas, and we will simply write $\models$ for the relation $\models^+$.

2.2 Expressive power

For an example of evaluating an **IFML**-formula, consider the modal structures $\mathfrak{M}$ and $\mathfrak{N}$ depicted in Figure 1. The atom p is true in $\mathfrak{M}$ precisely at c_1 and c_3, and in $\mathfrak{N}$ exactly at c_2'. Consider, then, the formula $\varphi := \Box(\Diamond/1)p$. We observe three things:

(1) φ is not true in $\mathfrak{M}$ at a: There is no winning strategy for $\exists$ in game $G(\varphi, \mathfrak{M}, a)$, since a function g inducing a winning strategy would have to satisfy $g(b_1) = g(b_2)$, and if this value was c_1 or c_3, the move would not be in accordance with the game rules if $\forall$'s choice was b_2 *resp.* b_1. On the other hand, if the value was c_2, the resulting plays would be wins for $\forall$, since p is not true at c_2. (As a matter of fact, φ is not false in $\mathfrak{M}$ at a either: also for $\forall$ there is no winning strategy in $G(\varphi, \mathfrak{M}, a)$. If $\forall$ chooses b_1 (b_2), then by choosing c_1 (*resp.* c_3) $\exists$ generates a play that she wins.)

(2) φ is true in $\mathfrak{N}$ at a': The function f defined by the condition $f(b_1') = f(b_2') = c_2'$ induces a winning strategy for $\exists$ in $G(\varphi, \mathfrak{N}, a')$.

(3) The structures $(\mathfrak{M}, a)$ and $(\mathfrak{N}, a')$ are bisimilar.[4] Hence they are not distinguished by any formula of basic modal logic.

In view of (1), (2) and (3), it follows that **IFML** is not translatable into **ML**. Since **ML** is trivially translatable into **IFML**, we have just established that **IFML** has greater expressive power than basic modal logic:[5]

[4] For bisimilarity, see, e.g., [BldRVe$_1$01, Section 2.2].
[5] This expressivity result was originally proven in [Tu$_1$03, Lemma 4].

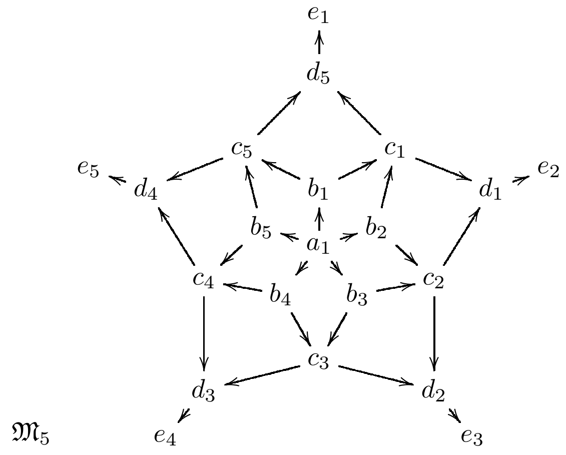


FIGURE 2.

Theorem 2.5. ML < IFML.

Our main result concerning the expressive power of **IFML** in this paper, Theorem 2.6, says that this logic is strong enough *not* to admit of a translation into first-order logic. This is in contradistinction to the case of basic modal logic, translatable via the well-known standard translation into **FO**, in fact into the 2-variable fragment of **FO**. (For standard translation, see, e.g., [BldRVe$_1$01, Section 2.4].)

Theorem 2.6. **IFML** is not translatable into **FO**.

Proof. Let $n \geq 2$ be arbitrary. In what follows, by stipulation $i \oplus 1 := i+1$, if $i < n$, and $n \oplus 1 := 1$. (Inversely, $i \ominus 1 = j$ means $j \oplus 1 = i$.) Define a modal structure $\mathfrak{M}_n = (M_n, R_n, V_n)$ as follows. The domain M_n consists of five disjoint layers, $L_0 := \{a_1\}$, $L_1 := \{b_1, \ldots, b_n\}$, $L_2 := \{c_1, \ldots, c_n\}$, $L_3 := \{d_1, \ldots, d_n\}$, and $L_4 := \{e_1, \ldots, e_n\}$, related by the accessibility relation $R_n :=$

$$\{(a_1, b_i) : 1 \leq i \leq n\} \cup \{(b_i, c_j) : 1 \leq j \leq n \text{ and } j \leq i \leq j \oplus 1\} \cup$$
$$\{(c_j, d_k) : 1 \leq k \leq n \text{ and } k \leq j \leq k \oplus 1\} \cup \{(d_k, e_{k\oplus 1}) : 1 \leq k \leq n\}.$$

The valuation V_n is empty. In Figure 2, the modal structure $\mathfrak{M}_5$ is depicted. Let, then, $\psi := \Box(\Box(\Diamond/2)(\Diamond/1,3)\top \vee \Box(\Diamond/2)(\Diamond/1,3)\top)$.

Claim 2.7. For all $n \geq 2$, $\mathfrak{M}_n, a_1 \models \psi$ if, and only if, n is even.

Proof. "From right to left": Suppose n is even. We define three functions, $f : L_1 \longrightarrow \{\texttt{left},\texttt{right}\}$, $g : \{\texttt{left},\texttt{right}\} \times L_1 \longrightarrow L_3$ and

$h : \{\texttt{left},\texttt{right}\} \times L_2 \longrightarrow L_4$. If $\forall$'s first move is b_i, define $f(b_i) = \texttt{left}$, if i is odd, and $f(b_i) = \texttt{right}$, otherwise. If $\forall$ continues by picking out c_j, put $g(\texttt{right}, b_i) = g(\texttt{left}, b_i) = d_{i\ominus 1}$, hence ignoring the information about c_j. Further, if j is odd and $f(b_i) = \texttt{left}$ (and so also i is odd), let $h(f(b_i), c_j) = e_j$, and similarly, if j is even and $f(b_i) = \texttt{right}$ (and so also i is even), let $h(f(b_i), c_j) = e_j$. Otherwise, let $h(f(b_i), c_j) = e_{j\oplus 1}$.

It is immediate that the functions f, g, h serve to define a winning strategy for $\exists$ in $G(\psi, \mathfrak{M}_n, a_1)$. In particular, when $\exists$ is supposed to make a choice corresponding to one of the occurrences of $(\Diamond/1, 3)$, she knows by f whether it is the right or the left disjunct that is at stake, and she sees $\forall$'s move c_j for the second occurrence of $\Box$ in that disjunction. She can infer whether $\forall$'s first choice was b_j or $b_{j\oplus 1}$, because by the evenness of n, the numbers j and $j \oplus 1$ cannot have the same parity, and having used f, $\exists$ has chosen the left disjunction if, and only if, j is odd. Knowing, then, which of the points b_j or $b_{j\oplus 1}$ $\forall$ had chosen, $\exists$ can further infer, by using g, at which point she currently is. But then there is only one point she can choose at all for $(\Diamond/1, 3)$, and this point is as a matter of fact given by h.

"From left to right": Assume n is odd, and suppose for contradiction that there is a winning strategy for $\exists$ in $G(\psi, \mathfrak{M}_n, a_1)$. Let f, g, h be functions as above induced by that winning strategy. Because n is odd, there necessarily are points $b_i, b_{i\oplus 1}$ such that $f(b_i) = f(b_{i\oplus 1})$. Let us w.l.o.g. assume that these points are b_n and b_1, and that $f(b_n) = f(b_1) = \texttt{left}$. Consider, then, the two partial plays where $\forall$'s pairs of choices are (b_n, c_n) and (b_1, c_n). The function g must yield for $(\Diamond/2)$ the choice d_{n-1} in the former case, and in the latter case the choice d_n. (Because of the uniformity condition, the choice must be the same no matter which successor of b_n *resp.* b_1 player $\forall$ chooses. In the former case, the options for $\forall$ are c_n and c_{n-1}, and their only common successor is d_{n-1}. And in the latter case $\forall$'s options are c_n and c_1, whose only common successor is d_n.)

The function h may only use as its arguments the disjunctive choice (which here is `left` in both cases) and $\forall$'s choice c_n—which likewise is the same for both partial plays, having (b_n, c_n, d_{n-1}) and (b_1, c_n, d_n) as their corresponding respective choices from the model. This means that h will choose the same point e_k in both cases. But whichever point e_k is, the move is possible along the accessibility relation R_n in at most one of the two cases. Hence f, g, h do not induce a winning strategy, contrary to the assumption. Q.E.D. (Claim 2.7)

Claim 2.8. For all $n \geq 1$, the first-order structures $\langle \mathfrak{M}^{\mathbf{FO}}_{2^n}, a_1 \rangle$ and $\langle \mathfrak{M}^{\mathbf{FO}}_{2^n+1}, a_1 \rangle$ satisfy exactly the same first-order formulas of one free variable and quantifier rank at most $n + 1$.

Proof. There is a winning strategy for *Duplicator* in the Ehrenfeucht-Fraïssé game $\mathbf{EF}_{n+1}(\mathfrak{M}^{\mathbf{FO}}_{2^n}, a_1; \mathfrak{M}^{\mathbf{FO}}_{2^n+1}, a_1)$: an optimal strategy for *Spoiler* is to choose successively the elements $b_{2^0}, b_{2^1}, \ldots, b_{2^n}$ from the domain of $\mathfrak{M}^{\mathbf{FO}}_{2^n+1}$; let *Duplicator* respond to these choices by the elements $b_{2^0}, b_{2^1}, \ldots, b_{2^n}$, respectively, from the domain of $\mathfrak{M}^{\mathbf{FO}}_{2^n}$. (Should *Spoiler* be allowed an $(n+2)$nd move, he could choose the element b_{2^n+1} from the domain of $\mathfrak{M}^{\mathbf{FO}}_{2^n+1}$, and to this *Duplicator* would have no response.) Q.E.D. (Claim 2.8)

In view of the Claims 2.7 and 2.8, ψ does not admit of translation into **FO**, and so **IFML** is not translatable into first-order logic.

Q.E.D. (Theorem 2.6)

Thus **IFML** is semantically a much stronger logic than **ML**, in fact, it cannot be translated into **FO**. It is an open question whether the satisfiability and validity problems of **IFML** are decidable.[6] In [Hy$_1$Tu$_1$05] the decidability of both of these problems was established for the so-called 'IF modal logic of perfect recall' ($\mathbf{IFML_{PR}}$). This logic is a fragment of **IFML**, syntactically restricted in such a way that the semantic games corresponding to its formulas are games of perfect recall. The structurally determined IF modal logic from Section 4 is likewise a fragment of **IFML**; like $\mathbf{IFML_{PR}}$, it is more expressive than **ML**; and its satisfiability and validity problems are decidable. The complexity of $\mathbf{IFML_{SD}}$-satisfiability is known to be in PSPACE [Tu$_1$Se$_2$06]; by contrast, the exact complexity of $\mathbf{IFML_{PR}}$-satisfiability is an open question (the recursive bound on the size of a finite model of an $\mathbf{IFML_{PR}}$-formula obtained in [Hy$_1$Tu$_1$05] has the form of tower function w.r.t. the length of the formula, and is hence far from feasible). The validity problems of the logics $\mathbf{IFML_{PR}}$ and $\mathbf{IFML_{SD}}$ are, on the other hand, both known to be decidable in PSPACE.

3 Extended IF modal logic

For one thing, the enterprise of IF logic teaches us that things that are uncontroversial and unproblematic in first-order logic turn out to have intriguing properties when we dare to introduce the slash device. A case in point is the behavior of propositional connectives, as observed by Hodges in [Ho$_1$97a].

Semantically, the evaluation of conjunction (disjunction) is a choice between two things: the left and the right conjunct (disjunct). Hence these connectives can be construed as restricted quantifiers. Instead of $(\psi \wedge \chi)$,

[6] Note that for IF modal logics, the satisfiability and validity problems are *not* each other's duals. Let '$\neg\psi$' be a shorthand notation for a formula in negation normal form such that: $\exists$ has a winning strategy in $G(\neg\psi, \mathfrak{M}, w)$ iff $\forall$ has a winning strategy in $G(\psi, \mathfrak{M}, w)$. Choose $\varphi, \mathfrak{M}$ and w so that φ is non-determined in $\mathfrak{M}$ at w. Then $(\varphi \vee \neg\varphi)$ also is non-determined in $\mathfrak{M}$ at w, and thus not valid. Yet $\neg(\varphi \vee \neg\varphi)$ is not satisfiable.

we may, equivalently, write $\bigwedge_{i\in\{\mathsf{L},\mathsf{R}\}}\varphi_i$, given that $\varphi_\mathsf{L} := \psi$ and $\varphi_\mathsf{R} := \chi$. Similar observations can be made about the restricted quantifier $\bigvee_{i\in\{\mathsf{L},\mathsf{R}\}}$. It is straightforward to see that in first-order logic, introducing these restricted quantifiers does not yield greater expressive power. E.g., the sentence $\wedge_{i\in\{\mathsf{L},\mathsf{R}\}}\exists x\ P_i x$ is simply equivalent to $(\exists x\ P_\mathsf{L} x \wedge \exists x\ P_\mathsf{R} x)$. The same holds for the result of replacing the usual conjunction and disjunction by the corresponding restricted quantifiers in basic modal logic: what results is just a notational variant of **ML**. For instance, $\wedge_{i\in\{\mathsf{L},\mathsf{R}\}}\Diamond_i p$ is equivalent to $(\Diamond_\mathsf{L} p \wedge \Diamond_\mathsf{R} p)$, where $\Diamond_i$ is the diamond over the accessibility relation R_i.

Hodges studied the restricted quantifiers $\bigwedge_{i\in\{\mathsf{L},\mathsf{R}\}}$ and $\bigvee_{i\in\{\mathsf{L},\mathsf{R}\}}$ in the context of IF logic in [Ho$_1$97a]. A similar move was made by Tulenheimo with respect to IF modal logic. The resulting logic, called *Extended IF modal logic*, was first introduced in [Tu$_1$04], having been originally suggested by Hyttinen (personal communication). This logic allows marking modal operators as independent even from superordinate conjunctions and disjunctions, when the latter are construed precisely as restricted quantifiers.

To make the *Eigenart* of Extended IF modal logic visible, we assume a polymodal framework. The modal structures considered will have k accessibility relations $R_1, \ldots, R_k$, each corresponding to a modality type of its own. In syntax, the diamonds and boxes carry an index, indicating which accessibility relation is responsible for the semantics of the operator in question. E.g., the formula $\Box_2\Diamond_7 p$ of a polymodal basic modal logic says that any R_2-successor of the current state has an R_7-successor satisfying the atom p.

Conjunctions and disjunctions are construed as restricted quantifiers ranging over the set $\{\mathsf{L},\mathsf{R}\}$. Accordingly, a string $i_1 \ldots i_n \in \{\mathsf{L},\mathsf{R}\}^*$ may appear as a subscript of a modal operator syntactically subordinate to n conjunction or disjunction signs, and it is a part of the specification of the syntax to associate the appropriate strings with modality types $1, \ldots, k$. For instance, if $\wedge_{i\in\{\mathsf{L},\mathsf{R}\}} \vee_{j\in\{\mathsf{L},\mathsf{R}\}} (\Diamond_{ij}/1)\top$ is a formula, the syntax must provide a mapping from the set $\{\mathsf{LL},\mathsf{LR},\mathsf{RL},\mathsf{RR}\}$ to the set $\{1, \ldots, k\}$. If the evaluation has proceeded to the subformula $(\Diamond_{ij}/1)\top$, the mapping yields a modality type corresponding to the diamond, depending on which choices among L,R were made earlier, first for $\wedge_{i\in\{\mathsf{L},\mathsf{R}\}}$ and then for $\vee_{j\in\{\mathsf{L},\mathsf{R}\}}$.

Before we get to the formal underpinnings of this logic, let us consider a nice illustration of its capabilities. Think of Extended IF modal logic with two modality types, one of which is interpreted by means of the identity relation $=$. Slightly abusing the syntax to make stating the example smoother, consider the formula $\wedge_{i\in\{=,R\}}(\Diamond_i/1)\top$, indicating that there is a state to which $\exists$ can move from the current state w, without knowing which accessibility relation (among R and $=$) was earlier picked out by $\forall$. One of the accessibility relations being $=$, $\exists$ must choose w. But this means that in order for this formula to hold in $\mathfrak{M}$ at w, it must also be possible to get from w to w via the relation R. In fact, when evaluated at w, the formula serves to state that $(w, w) \in R$.

Syntax. Let $L(\mathbf{prop})$ be the set of literals: formulas of the form p or $\neg p$ with $p \in \mathbf{prop}$. *Extended IF modal logic*, $\mathbf{EIFML}_k$, uses k modality types, and its formulas are strings $O_1 \ldots O_n \gamma(j_1 \ldots j_m)$, where m is the total number of conjunction and disjunction symbols in the prefix $O_1 \ldots O_n$. The components of these strings are as follows. First, the strings are associated with a *distribution of modality types* $\mu : \bigcup_{i \leq m} \{\mathsf{L}, \mathsf{R}\}^i \longrightarrow \{1, \ldots, k\}$ and a *distribution of literals* $\gamma : \{\mathsf{L}, \mathsf{R}\}^m \longrightarrow L(\mathbf{prop})$. Second, each O_x is one of the following:

(i) $\bigwedge_{j_x \in \{\mathsf{L},\mathsf{R}\}}$;

(ii) $\bigvee_{j_x \in \{\mathsf{L},\mathsf{R}\}}$;

(iii) $\Box_{j_1 \ldots j_y}$, where y is the number of conjunction and disjunction symbols preceding O_x in the prefix;

(iv) $(\Diamond_{j_1 \ldots j_y} / i_1, \ldots, i_z)$, where y is the number of conjunction and disjunction symbols preceding O_x in the prefix, and $1 \leq i_1, \ldots, i_z \leq x - 1$.

Semantics. The semantics will be defined relative to k-ary modal structures, mentioned in Section 1. To formulate the truth conditions, a semantic game $G(\varphi, \mathfrak{M}, w_0)$ is associated with each formula φ, k-ary modal structure $\mathfrak{M}$ and state $w_0 \in M$. In the interest of clarity, we will define positions in the game so that they keep explicitly track not only of the states chosen before reaching that position, but also of the conjuncts and disjuncts chosen up to then.

Definition 3.1. Let $\varphi := O_1 \ldots O_n \gamma(j_1 \ldots j_m) \in \mathbf{EIFML}_k$. Positions of game $G(\varphi, \mathfrak{M}, w_0)$ are quadruples $(\psi, \ell, \varsigma, \varsigma')$, where $1 \leq \ell \leq n+1$, $\psi = O_\ell \ldots O_n \gamma(j_1 \ldots j_m)$, $\varsigma : S \longrightarrow M$ and $\varsigma' : S' \longrightarrow \{\mathsf{L}, \mathsf{R}\}$, where S is the set of those numbers x in $\{1, \ldots, n\}$ for which O_x is a modal operator, and $S' = \{1, \ldots, n\} \backslash S$. (The functions ς, ς' may simply be thought of as sequences of states and sequences of objects L, R, respectively.) In the beginning, the position is $(\varphi, 1, \langle w_0 \rangle, \varnothing)$. The following conditions generate the set of all positions of $G(\varphi, \mathfrak{M}, w_0)$, with $\mathfrak{M} = (M, R_1, \ldots, R_k, V)$. They also specify which player makes which type of choice (if any) at a given position.

1. *(a)* If $\gamma(\varsigma') = p$ and the position is $(p, n+1, \varsigma, \varsigma')$, then: if $\max(\varsigma) \in V(p)$, $\exists$ wins, else $\forall$ wins. *(b)* If $\gamma(\varsigma') = \neg p$ and the position is $(\neg p, n+1, \varsigma, \varsigma')$, then: if $\max(\varsigma) \notin V(p)$, $\exists$ wins, otherwise $\forall$ wins.

2. If the position is $(\bigvee_{j_x \in \{\mathsf{L},\mathsf{R}\}} \varphi, \ell, \varsigma, \varsigma')$, then both $(\varphi, \ell+1, \varsigma, \varsigma'{}^\frown \mathsf{L})$ and $(\varphi, \ell+1, \varsigma, \varsigma'{}^\frown \mathsf{R})$ are positions. Player $\exists$ chooses one of these positions at $(\bigvee_{j_x \in \{\mathsf{L},\mathsf{R}\}} \varphi, \ell, \varsigma, \varsigma')$. The case of $(\bigwedge_{j_x \in \{\mathsf{L},\mathsf{R}\}} \varphi, \ell, \varsigma, \varsigma')$ is otherwise similar, but it is player $\forall$ who chooses one of the positions at $(\bigwedge_{j_x \in \{\mathsf{L},\mathsf{R}\}} \varphi, \ell, \varsigma, \varsigma')$.
3. If $\mu(\varsigma') = j$, and the position is $((\Diamond_{\varsigma'}/i_1, \ldots, i_z)\varphi, \ell, \varsigma, \varsigma')$, then for every v such that $\langle \max(\varsigma), v\rangle \in R_j$, we have that $(\varphi, \ell+1, \varsigma^\frown v, \varsigma')$ is a position. It is $\exists$ who chooses one of these, or if none exists, there are no further positions and $\forall$ wins. The case of $(\Box_{\varsigma'}\varphi, \ell, \varsigma, \varsigma')$ is similar, but it is $\forall$ who makes the choice, or, if he can make none, there are no further positions and $\exists$ wins.

The notions of game tree and (partial) play are defined as in the case of **IFML**. A strategy of $\exists$ is also defined similarly, as a tuple of strategy functions $\sigma = (\sigma_1, \ldots, \sigma_h)$, with one strategy function for each expression of the form $\bigvee_{j_x \in \{\mathsf{L},\mathsf{R}\}}$ or $(\Diamond_{j_1\ldots,j_y}/i_1, \ldots, i_z)$ in the prefix. If a partial play $(p_0, \ldots, p_{i-1})$ is already produced, such a strategy function makes a choice between the positions each of which is a combinatorially possible next position. Using a strategy is again defined like in the case of **IFML**. A strategy of $\exists$ is winning, if it leads to a win against any sequence of moves by $\forall$, and is, furthermore, uniform in the following sense: if $p_i = (\varphi, \ell, \varsigma, \varsigma')$ and $p_i' = (\varphi, \ell, \tau, \tau')$ are any two positions arising in the game supposing that $\exists$ has used σ, with $\varphi = (\Diamond_{j_1\ldots,j_y}/i_1, \ldots, i_z)\psi$, then if the maps $\varsigma \cup \varsigma'$ and $\tau \cup \tau'$ agree on all their values except possibly on $i_1, \ldots, i_z$, the strategy σ agrees on the positions p_i and p_i', i.e., maps the sequence of positions leading to p_i to the same element as the sequence of positions leading to p_i'. The notions of strategy, using a strategy and winning strategy are defined analogously for player $\forall$ (keeping in mind that the condition of uniformity is vacuously satisfied by $\forall$'s strategies).

Semantics of $\mathbf{EIFML}_k$ is simply defined by stipulating that φ is true (false) in $\mathfrak{M}$ at w_0, if there is a winning strategy for $\exists$ (*resp.* $\forall$) in game $G(\varphi, \mathfrak{M}, w_0)$.

About the expressiveness of Extended IF modal logic, note first that by the proof of Theorem 2.6, $\mathbf{EIFML}_1$ cannot be translated into **FO**.[7] Hence:

Theorem 3.2. For all $k \geq 1$, $\mathbf{EIFML}_k$ is not translatable into **FO**.

Second, it is evident that polymodal $\mathbf{EIFML}_k$ is more expressive than the polymodal version of **IFML**. E.g., consider evaluating the formula $\wedge_{i \in \{=,R\}}(\Diamond_i/1)\top$ of our earlier example relative to the modal structures $(\{a\}, \{(a,a)\}, \{(a,a)\}, \varnothing)$ and $(\{a,b\}, \{(a,a)\}, \{(a,b)\}, \varnothing)$ with $a \neq b$.

[7] This considerably improves the result proven in [Tu$_1$04, Theorem 3.3.9], according to which, whenever $k \geq 3$, $\mathbf{EIFML}_k$ does not have a first-order translation.

Among the applauded virtues of modal logic are its nice computational properties: for instance, model checking is tractable and satisfiability is decidable. Amusingly, it can be shown that the satisfiability problem of $\mathbf{EIFML}_k$ with the identity relation is undecidable, cf. [Se$_2$06]. This result shows that the power of slashing is considerable even when we import it in modal logic. It is interesting to see whether a similar undecidability result can be achieved without the identity relation and maybe even for **IFML**. It might well turn out that **IFML** proper is decidable, whereas $\mathbf{EIFML}_k$ is undecidable. This would show us—once again—that the particulars of a pre-slash, independence-unfriendly language are sleeping beauties.

4 Structurally determined IF modal logic

The logics **IFML** and $\mathbf{EIFML}_k$ aimed at being modal analogues of IF first-order logic: results of importing the slash device into modal logic and interpreting it so as to produce a modal logic of informational independence. However, as indeed shown by the two languages, the particular independence friendly modal logic that we end up considering depends highly on the syntax used, and on the way independence is introduced. In Section 4.1, we introduce a new framework in which IF modal logics can be compared and isolated, by tuning three parameters that will be highlighted shortly. Some researchers have objected that there is no *a priori* reason why slashed modal operators would formalize a meaningful notion of independence. This criticism will be revisited in relation to the logics specified within our framework. Section 4.2 singles out a specific logic by instantiating the three parameters of the framework in a certain way. The logic in question will be a fragment of IF first-order logic; it is denoted by $\mathbf{IF}(\mathrm{ST}^2(\mathbf{ML}))$. Section 4.3 introduces a certain modal-like logic—so-called 'structurally determined IF modal logic'—which is subsequently, in Section 4.4, shown to characterize $\mathbf{IF}(\mathrm{ST}^2(\mathbf{ML}))$. Interestingly, this modal-like logic will have a compositional, 'Tarskian' semantics. Finally, Section 4.5 discusses some aspects of the expressive power of the structurally determined IF modal logic.

4.1 The framework

The framework we propose essentially isolates 'modal fragments' of IF first-order logic. This approach is partially inspired by current research in modal logic. Namely, although basic modal logic is an extension of propositional logic, nowadays it is usually conceived of as a fragment of first-order logic. Milestone results that brought about this change of perspective include the standard translation, and van Benthem's Theorem [vB76] which characterizes modal logic as the '*bisimulation invariant*' fragment of first-order logic.

Assuming this perspective, an IF modal logic is obtained by fixing three things: first, a set of strings that are considered as modal formulas; second, a standard translation that maps the former set to a subset of first-order logic; and third, an IF procedure that maps this subset of first-order logic into IF first-order logic. The IF modal logic thus generated is *modal* in that it originates from a modal logic, and *independence friendly* in that it is a fragment of IF first-order logic, through the appropriate IF procedure.

More precisely, our framework covers all logics that are obtained from a modal logic $\mathcal{ML}$ that can be translated into first-order logic, standard translation ST and IF procedure **IF** as follows:

- Translate every formula from $\mathcal{ML}$ into first-order logic using ST, and obtain the first-order correspondence language of $\mathcal{ML}$, denoted $\mathrm{ST}(\mathcal{ML})$.
- Apply to every formula from $\mathrm{ST}(\mathcal{ML})$ the IF procedure **IF**, and obtain the set of IF modal formulas determined by $\mathcal{ML}$, ST, and **IF**, denoted $\mathbf{IF}(\mathrm{ST}(\mathcal{ML}))$.

Observe that there are instantiations of the initial modal logic, the standard translation, and the IF procedure that give rise to meaningless and uninteresting systems. This will happen if the parameters $\mathcal{ML}$, ST and **IF** are incompatible; for instance, if the range of the operation ST is disjoint from the domain of the operation **IF**. But the framework also contains potentially interesting systems. For one thing, as we will see, the (IF first-order correspondence languages of the) logics that were studied earlier under the headings **IFML** and $\mathbf{EIFML}_k$ can be generated by tuning the parameters of the framework in a specific way (cf. Table 1 in Section 4.2).

We think this framework facilitates finding logics that have 'nice' combinations of properties. It is beyond the scope of the current paper to give a specific sense to the phrase 'nice combination of properties'. But generally a high expressive power combined with a low computational complexity is considered nice. In the context of IF logic, also allowing for a compositional semantics can be appreciated as a desirable property.

Indeed, Cameron and Hodges showed in [Ca$_1$Ho$_1$01] that *no* compositional semantics exists for IF first-order logic in which the 'interpretation' $|\varphi(x)|_\mathcal{A}$ of a formula with one free variable is a *subset* of the domain of $\mathcal{A}$; what is more, they even proved that in a compositional semantics for IF first-order logic, $|\varphi(x)|_\mathcal{A}$ cannot be a subset of $\mathrm{dom}(\mathcal{A})^n$, for any $n < \omega$. If by 'Tarskian semantics' we mean a compositional semantics where the interpretation of a formula $|\varphi(x_1, \ldots, x_k)|_\mathcal{A}$ will be a subset of $\mathrm{dom}(\mathcal{A})^m$ for some $m \geq k$, it follows that no Tarskian semantics for IF first-order logic is possible. (On the other hand, Hodges had already proven in [Ho$_1$97a, Ho$_1$97b] that IF first-order logic admits of a compositional semantics where the interpretation $|\varphi(x)|_\mathcal{A}$ of each formula $\varphi(x)$ is a *subset of the powerset* of the domain.) Given this background, being able to show that an IF modal logic can be interpreted in a Tarskian way, signals that the complexity of the full IF logic is tamed in this respect.

As an example of a logic emerging from our framework, we will consider the structurally determined IF modal logic introduced in [Tu$_1$Se$_2$06]. The satisfiability and validity problems of this logic are decidable, and it allows for a compositional semantics.

Some researchers have opposed the very idea of an independence friendly modal logic, by insisting that independence is a relation between (syntactically manifest) variables, while none are forthcoming in modal logics. Thus, they have suggested that *a priori* modal operators furnished with independence indications are not necessarily meaningful. Staying within our framework, we need not enter such a discussion. Rather, we can point out that logics generated in our framework are immune to any such criticism, since they are literally fragments of IF first-order logic.

4.2 Instantiating the three parameters

In this subsection, we will fix the values of the parameters in a certain way. It turns out that the resulting fragment of IF first-order logic admits of a particularly smooth characterization in modal logical terms (see Section 4.3). Actually, it is captured by a compositional modal-like language, to be referred to as $\mathbf{IFML_{SD}}$. The concrete cases we wish to consider are these:

- Basic modal logic, **ML**, in *negation normal form.* (It is well known that each basic modal formula has an equivalent form in which the negation-sign appears only as prefixed to a propositional atom.)
- The standard translation $\mathrm{ST}^2 : \mathbf{ML} \longrightarrow \mathbf{FO}^2$ of basic modal logic into the 2-variable fragment of first-order logic.
- The IF procedure associating with every first-order formula φ the set $\mathbf{IF}(\varphi)$ of those IF first-order formulas that are obtained by replacing any number of existential quantifiers $\exists x_k$ appearing in φ by the corresponding symbol $(\exists x_k/x_{i_1}, \ldots, x_{i_n})$, provided that: (a) in φ there appear the universal quantifiers $\forall x_{i_1}, \ldots, \forall x_{i_n}$ superordinate to $\exists x_k$; (b) in the formula resulting from the replacement, the variables $x_{i_1}, \ldots, x_{i_n}$ in $(\exists x_k/x_{i_1}, \ldots, x_{i_n})$ become thereby bound by the universal quantifiers $\forall x_{i_1}, \ldots, \forall x_{i_n}$; and (c) $x_k \notin \{x_{i_1}, \ldots, x_{i_n}\}$.

Note that clause (b) precludes cases like the string $\forall x \exists x (\exists y/x)\varphi$, resulting from applying the IF procedure to $\forall x \exists x \exists y \varphi$.

Basic modal logic is assumed to be in negation normal form (NNF), to ensure that its first-order correspondence language is in negation normal form as well. This is useful in the present context, since one may safely apply a Hintikka-style IF procedure to extend any fragment of **FO** that is in NNF. Another way in which a basic modal language interesting for the purposes of IF modal logic can be introduced is using the strong prenex normal form (SPNF), by now familiar from $\mathbf{EIFML}_k$: considering formulas $O_1 \dots O_n \gamma(j_1 \dots j_m)$, where each O_x in the prefix is $\bigwedge_{j_x \in \{\mathsf{L},\mathsf{R}\}}$, $\bigvee_{j_x \in \{\mathsf{L},\mathsf{R}\}}$, $\Box_{j_1 \dots j_y}$ or $\Diamond_{j_1 \dots j_y}$ (y being the number of conjunction and disjunction symbols preceding O_x in the prefix), and for all strings $j_1 \dots j_m \in \{\mathsf{L},\mathsf{R}\}^m$, we have that $\gamma(j_1 \dots j_m)$ is a literal. As was already noted in Section 2.2, the standard translation of **ML** into **FO** can be performed using only two variables. Of course, other standard translations mapping **ML** into **FO** are readily available, say one that introduces pairwise distinct variables for any quantifiers translating nested modal operators. Finally, as remarked earlier, various IF procedures have been proposed. The one put forward by us is tailor-made to suit the particulars of the fragment of first-order logic obtained by translating **ML** to **FO** via ST^2.

By stipulation the two variables of $\mathbf{FO}^2$ are x, y. Define the *2-variable fragment of IF first-order logic*, denoted $\mathbf{IF}^2$, as follows. Its formulas are obtained from those of $\mathbf{FO}^2$: Let $\alpha, \beta \in \{x, y\}$ and $\alpha \neq \beta$. If $\varphi \in \mathbf{FO}^2$, the result of replacing any number of occurrences of $\exists\beta$ subordinate to $\forall\alpha$ in φ by the symbol $(\exists\beta/\alpha)$ is a formula of $\mathbf{IF}^2$, provided that the variable β in $(\exists\beta/\alpha)$ becomes thereby bound by $\forall\alpha$. Thus $\exists x \forall x (\exists y/x) Rxy$ is a formula of $\mathbf{IF}^2$, but $\forall x \exists x (\exists y/x) Rxy$ is not. The semantics of $\mathbf{IF}^2$ is obtained from the semantics of IF first-order logic by stipulating that the variable α mentioned in $(\exists\beta/\alpha)$ is bound by the *closest* universal quantifier $\forall\alpha$ *superordinate* to $(\exists\beta/\alpha)$.

Having made the three choices and defined $\mathbf{IF}^2$, a fragment of IF first-order logic, to be denoted $\mathbf{IF}(\mathrm{ST}^2(\mathbf{ML}))$, has been determined. It consists of the results of applying the specified IF procedure to the first-order formulas yielded by the standard translation ST^2 from the formulas of basic modal logic in NNF. The framework of structurally determining a logic is well-suited also for discussing other IF modal logics. Table 1 lists (the IF first-order correspondence languages of) several IF modal logics studied in the literature, in terms of different instantiations of the three parameters.

The logics L_1, L_2, L_3 are the IF first-order correspondence languages of $\mathbf{IFML_{PR}}$, **IFML** *resp.* $\mathbf{EIFML}_k$, i.e., the canonical translations of these IF modal logics into the suitable formulation of IF first-order logic. L_4 is the logic $\mathbf{IF}(\mathrm{ST}^2(\mathbf{ML}))$. The standard translation needed for the logics L_1, L_2 and L_3 introduces distinct quantified variables for any two nested modal operators. The standard translation of L_3 further construes propositional connectives as restricted quantifiers.

4.3 Structurally determined IF modal logic

Our framework generates fragments of IF first-order logic. They may be hard to parse. Therefore we aim at characterizing the logic $\mathbf{IF}(\mathrm{ST}^2(\mathbf{ML}))$

	Basic mod. log. in	*Translation into*	*$\exists x_i$ can be indep. of*
L_1	NNF	**FO** in NNF	$\forall x_j$ if betw. $\forall x_j$ and $\exists x_i$, no $\vee$ or $\exists x_k$ appears
L_2	NNF	**FO** in NNF	$\forall x_j$ or $\exists x_j$
L_3	SPNF	**FO** in SPNF	$\forall x_j$, $\exists x_j$, $\wedge$ or $\vee$
L_4	NNF	$\mathbf{FO}^2$ in NNF	$\forall x_j$ if $x_i \neq x_j$ and betw. $\forall x_j$ and $\exists x_i$, $\exists x_j$ does not appear

TABLE 1. Results of different instantiations of the three parameters.

in terms of a more transparent, modal machinery. We wish to structurally determine a modal logic—'IF modal logic' with a modal syntax—by singling it out as the logic **X** such that its translation $\mathrm{ST}^{\mathbf{IF}}(\mathbf{X})$ into the 2-variable fragment $\mathbf{IF}^2$ of IF first-order logic coincides with the result of applying the IF procedure ($\Downarrow_{\mathrm{IF}}$) to the $\mathbf{FO}^2$-translation of **ML**:

$$\begin{tikzcd}
\mathbf{ML} \arrow[r, "\mathrm{ST}^2"] \arrow[d, phantom, "\cap"] & \mathrm{ST}^2(\mathbf{ML}) \arrow[r, phantom, "\subset"] \arrow[d, Rightarrow, "\mathrm{IF}"] & \mathbf{FO}^2 \arrow[d, Rightarrow, "\mathrm{IF}"] \\
{} & \mathbf{IF}(\mathrm{ST}^2(\mathbf{ML})) \arrow[r, phantom, "\subset"] \arrow[d, phantom, "\shortparallel"] & \mathbf{IF}^2 \\
\mathbf{X} \arrow[r, "\mathrm{ST}^{\mathbf{IF}}"] & \mathrm{ST}^{\mathbf{IF}}(\mathbf{X}) &
\end{tikzcd}$$

In want of better terminology, we will refer to the language **X** as 'structurally determined IF modal logic' (and will denote it by $\mathbf{IFML_{SD}}$).

As will be shown in this subsection, we will be able to find a particularly nice modal-like presentation for the IF modal logic **X**: a presentation as a modal-like logic with a compositional semantics. We do *not* wish to suggest that this would be an integral part of our proposed framework of structurally determining modal logics. E.g., for **IFML** and $\mathbf{EIFML}_k$, we do not have such a Tarskian compositional characterization, and neither are we aware of a possibility of obtaining one (except by formulating a non-Tarskian modal 'trump semantics', i.e., by doing to the relevant IF modal logics what Hodges did in [Ho$_1$97a] to IF first-order logic).

Let us define the syntax and semantics of a modal-like logic, which turns out to be the logic **X** structurally determined above (see Proposition 4.2).

Syntax. A class of formulas is generated by the two grammars A and B:

$$\begin{array}{lcl}
\alpha & ::= & p \mid \neg p \mid (\alpha \vee \alpha) \mid (\alpha \wedge \alpha) \mid \Diamond\alpha \mid \Box\alpha \mid \blacksquare\beta \\
\beta & ::= & \blacklozenge\alpha \mid (\alpha \vee \beta) \mid (\beta \vee \alpha) \mid (\beta \vee \beta) \mid (\alpha \wedge \beta) \mid (\beta \wedge \alpha) \mid (\beta \wedge \beta),
\end{array}$$

where $p \in$ **prop**. The two grammars generate the formulas of a logic we refer to as $\mathbf{IFML}^{\circ}_{\mathbf{SD}}$. The formulas α are said to be *closed*, and the formulas

β open. If φ is a formula, all tokens of $\blacklozenge$ not subordinate to a token of $\blacksquare$ in φ are called *free*. If φ is open, all its free tokens of $\blacklozenge$ become *bound by* the outmost token of $\blacksquare$ in $\blacksquare\varphi$. For instance, $\blacklozenge p$ is open; and $\blacksquare(\blacklozenge p \vee \blacklozenge q)$ is closed, the two tokens of $\blacklozenge$ being bound by the unique token of $\blacksquare$. We stipulate that the formulas of the *structurally determined IF modal logic*, $\mathbf{IFML_{SD}}$, are the closed formulas of $\mathbf{IFML^{\circ}_{SD}}$ (i.e., the formulas generated by the grammar A). Note that the reason why the grammar B contains both clauses $(\alpha \circ \beta)$ and $(\beta \circ \alpha)$ with $\circ \in \{\wedge, \vee\}$ is simply 'aesthetic': we wish that the conjunctions and disjunctions of formulas of which one is open and the other is closed, can be formed in either order.

The operators $\blacksquare$ and $\blacklozenge$ will be referred to as 'black box' and 'black diamond', and the operators $\Box$ and $\Diamond$ as 'white box' and 'white diamond'. Intuitively, $\blacklozenge$ is the 'independent diamond', and it will by definition be independent precisely of the token of $\blacksquare$ that binds it. For its part, this logic will hence illustrate that the relations 'being bound by' and 'being logically dependent on' need not coincide; this point is made in a more general context by Hintikka [Hi$_1$97].

Semantics. For every $\varphi \in \mathbf{IFML^{\circ}_{SD}}$, a satisfaction relation

$$\mathfrak{M}, I, \bar{\imath}, w \models \varphi$$

is defined, where $\mathfrak{M} = (M, R, V)$ is a modal structure and $w \in M$ is a state as usual, and furthermore, $I : \{0,1\}^* \longrightarrow M$ is a *token valuation*, and $\bar{\imath}$ is a binary string: $\bar{\imath} \in \{0,1\}^*$. Here, 0 and 1 should be intuitively thought of as the choices *left* and *right*, respectively, made when interpreting propositional connectives $(\wedge, \vee)$. Observe that given a formula φ, a binary string $i_1 \ldots i_n$ determines a subformula ψ of φ, namely the formula yielded by starting to go through, outside-in, the propositional connectives of φ and choosing for the j-th connective encountered *left* or *right* according to whether i_j is 0 or 1. The process stops either because there are no more connectives to go or because all the n choices have been made. The determined formula ψ is the subformula reached by the process.

In providing the semantics of formulas of the form $\blacklozenge\varphi$, the token valuation I will be used. The idea is that I will yield states interpreting particular tokens of $\blacklozenge$, the tokens being identified precisely in terms of binary strings $\bar{\imath} \in \{0,1\}^*$. Hence, in particular, the state interpreting the token of $\blacklozenge$ prefixing $\blacklozenge\varphi$ is determined by I; and in general the valuation I has been chosen already earlier in the evaluation (namely, when interpreting the closest superordinate $\blacksquare$), so that the state to interpret the token of $\blacklozenge$ in question has been determined, as it were, in advance. The truth conditions of literals and formulas of the forms $\Box$ and $\Diamond$ do not make use of the components I and $\bar{\imath}$ of the models; by contrast, the components I and $\bar{\imath}$ play a key role in the rest of the clauses:

$$\begin{array}{rll}
\mathfrak{M}, I, \bar{\imath}, w \models p & \text{iff} & w \in V(p) \\
\mathfrak{M}, I, \bar{\imath}, w \models \neg p & \text{iff} & w \notin V(p) \\
\mathfrak{M}, I, \bar{\imath}, w \models \Diamond\varphi & \text{iff} & \text{for some } v \text{ with } R(w,v)\text{: } \mathfrak{M}, I, \bar{\imath}, v \models \varphi \\
\mathfrak{M}, I, \bar{\imath}, w \models \Box\varphi & \text{iff} & \text{for every } v \text{ with } R(w,v)\text{: } \mathfrak{M}, I, \bar{\imath}, v \models \varphi \\
\mathfrak{M}, I, \bar{\imath}, w \models (\psi \vee \varphi) & \text{iff} & \mathfrak{M}, I, \bar{\imath}0, w \models \psi \ \text{ or } \ \mathfrak{M}, I, \bar{\imath}1, w \models \varphi \\
\mathfrak{M}, I, \bar{\imath}, w \models (\psi \wedge \varphi) & \text{iff} & \mathfrak{M}, I, \bar{\imath}0, w \models \psi \ \text{ and } \ \mathfrak{M}, I, \bar{\imath}1, w \models \varphi \\
\mathfrak{M}, I, \bar{\imath}, w \models \blacksquare\varphi & \text{iff} & \text{for some } I' : \{0,1\}^* \longrightarrow M\text{: } \mathfrak{M}, I', \bar{\imath}, w \models \Box\varphi \\
\mathfrak{M}, I, \bar{\imath}, w \models \blacklozenge\varphi & \text{iff} & R(w, I(\bar{\imath})) \text{ and } \mathfrak{M}, I, \bar{\imath}, I(\bar{\imath}) \models \varphi.
\end{array}$$

It should be observed that for the token valuations in $\mathbf{IFML}^{\circ}_{\mathbf{SD}}$-semantics, what really matters are the *free* occurrences of the black diamond in a formula. It can easily be checked that if (relative to an initial string $\bar{\imath}$) the free occurrences of $\blacklozenge$ in φ are those identified by the strings in the set $S \subset \{0,1\}^*$ (which is necessarily finite), then if for *some* I, we have $\mathfrak{M}, I, \bar{\imath}, w \models \varphi$, actually $\mathfrak{M}, I', \bar{\imath}, w \models \varphi$ holds for any I' such that for all $\bar{\jmath} \in S$, $I'(\bar{\jmath}) = I(\bar{\jmath})$. In particular, the satisfaction condition does not require that we have $I'(\bar{\imath}) = I(\bar{\imath})$, unless $\bar{\imath} \in S$. It follows that the semantic clause for the black diamond need not be phrased in terms of quantification over token valuations: to evaluate $\blacksquare\varphi$, it suffices to choose a fixed finite number of states: as many states as there are free occurrences of $\blacklozenge$ in φ to be interpreted. Let us write $\mathfrak{M}, w \models \varphi$ to express the following condition: for all token valuations $I : \{0,1\}^* \longrightarrow M$ and all strings $\bar{\imath} \in \{0,1\}^*$, we have $\mathfrak{M}, I, \bar{\imath}, w \models \varphi$. Now note that if the $\mathbf{IFML}^{\circ}_{\mathbf{SD}}$-formula φ is closed (i.e., if $\varphi \in \mathbf{IFML}_{\mathbf{SD}}$) and for *some* I and $\bar{\imath}$, we have $\mathfrak{M}, I, \bar{\imath}, w \models \varphi$, then actually $\mathfrak{M}, w \models \varphi$ holds. Formulas of $\mathbf{IFML}_{\mathbf{SD}}$ are in this respect like *sentences* in first-order logic: if satisfied under one assignment, they are satisfied under all assignments. Their being satisfied is entirely independent of the assignment. By contrast, for open $\mathbf{IFML}^{\circ}_{\mathbf{SD}}$-formulas the token valuations and the binary strings have a crucial relevance. Formally, free tokens of $\blacklozenge$ (as identified by certain strings) bear resemblance to free variables, and the valuations to variable assignments; in the presence of free variables the satisfaction conditions of first-order logic are of course essentially dependent on the assignments.

4.4 Standard translation

In this subsection we show that the modal-like logic $\mathbf{IFML}_{\mathbf{SD}}$ is equally expressive as the logic $\mathbf{IF}(\mathrm{ST}^2(\mathbf{ML}))$ specified in Section 4.2. This result is interesting, because it shows that this fragment of IF first-order logic indeed can be given a Tarskian semantics, in the sense specified in Section 4.1, unlike the full IF first-order logic. To be precise, in the compositional semantics we designed for $\mathbf{IFML}^{\circ}_{\mathbf{SD}}$, the interpretation $|\varphi|_{\mathfrak{M}}$ of a formula φ in a modal structure $\mathfrak{M} = (M, R, V)$ is, in effect, a set of $(n+1)$-tuples of elements of M, where n is the number of *free tokens of the black diamond* in

φ, the remaining member of the tuple simply specifying the state w relative to which the formula is evaluated. Hence the analogue of [Ca$_1$Ho$_1$01, Cor. 6.2] does not hold for the logic $\mathbf{IFML_{SD}}$: we need not climb to the level of the powerset of the domain to obtain a compositional semantics; for any formula of $\mathbf{IFML^{\circ}_{SD}}$, a fixed Cartesian power of the domain suffices.

Basic modal logic has a translation into the 2-variable fragment of first-order logic. The class of closed $\mathbf{IFML^{\circ}_{SD}}$-formulas, that is, $\mathbf{IFML_{SD}}$, has an analogous property. Concretely, the following map $\mathrm{ST}^{\mathbf{IF}}_x : \mathbf{IFML_{SD}} \longrightarrow \mathbf{IF}^2$ provides a canonical translation of $\mathbf{IFML_{SD}}$ into the 2-variable fragment of IF first-order logic. For all $\alpha, \beta \in \{x, y\}$ with $\alpha \neq \beta$, define:

$$\begin{aligned}
\mathrm{ST}^{\mathbf{IF}}_\alpha(p) &= P\alpha,\\
\mathrm{ST}^{\mathbf{IF}}_\alpha(\neg p) &= \neg P\alpha,\\
\mathrm{ST}^{\mathbf{IF}}_\alpha((\varphi \circ \psi)) &= (\mathrm{ST}^{\mathbf{IF}}_\alpha(\varphi) \circ \mathrm{ST}^{\mathbf{IF}}_\alpha(\psi)) \quad \text{if } \circ \in \{\vee, \wedge\},\\
\mathrm{ST}^{\mathbf{IF}}_\alpha(\Diamond\varphi) &= \exists\beta(R\alpha\beta \wedge \mathrm{ST}^{\mathbf{IF}}_\beta(\varphi)),\\
\mathrm{ST}^{\mathbf{IF}}_\alpha(\Box\varphi) &= \forall\beta(R\alpha\beta \rightarrow \mathrm{ST}^{\mathbf{IF}}_\beta(\varphi)),\\
\mathrm{ST}^{\mathbf{IF}}_\alpha(\blacklozenge\varphi) &= (\exists\beta/\alpha)(R\alpha\beta \wedge \mathrm{ST}^{\mathbf{IF}}_\beta(\varphi)),\\
\mathrm{ST}^{\mathbf{IF}}_\alpha(\blacksquare\varphi) &= \mathrm{ST}^{\mathbf{IF}}_\alpha(\Box\varphi).
\end{aligned}$$

Clearly, if φ is a closed $\mathbf{IFML^{\circ}_{SD}}$-formula, then $\mathrm{ST}^{\mathbf{IF}}_x(\varphi)$ is an $\mathbf{IF}^2$-formula with exactly one free variable, x. The map $\mathrm{ST}^{\mathbf{IF}}_x$ provides a translation:

Proposition 4.1. For every formula $\varphi \in \mathbf{IFML_{SD}}$, modal structure $\mathfrak{M}$, and state w: $\mathfrak{M}, w \models \varphi$ iff $\mathfrak{M}^{\mathbf{FO}}, w \models \mathrm{ST}^{\mathbf{IF}}_x(\varphi)$.

Proof. The proposition can be proven by induction on the structure of closed formulas. Observe that the formulas prefixed with $\blacksquare$ are of the form $\blacksquare\psi$, where ψ is obtained by conjunction and disjunction from closed formulas and formulas of the form $\blacklozenge\theta$, where θ is closed. Q.E.D.

Further, the following 'commutativity' result holds, establishing that $\mathbf{IFML_{SD}}$ actually is the logic $\mathbf{X}$ structurally determined above:

Proposition 4.2. Syntactically, $\mathrm{ST}^{\mathbf{IF}}_x(\mathbf{IFML_{SD}}) = \mathbf{IF}(\mathrm{ST}^2_x(\mathbf{ML}))$.

Proof. The inclusion from left to right: Let $\varphi \in \mathbf{IFML_{SD}}$ be arbitrary, and let φ^- be the $\mathbf{ML}$-formula resulting from φ by turning all its black boxes and black diamonds into their white counterparts. Clearly $\mathrm{ST}^{\mathbf{IF}}_x(\varphi)$ is obtained by the IF procedure from $\mathrm{ST}^2_x(\varphi^-)$. *The inclusion from right to left:* Let $\psi \in \mathbf{ML}$ be arbitrary, and let ψ^+ be any result of applying the IF procedure to $\mathrm{ST}^2_x(\psi)$. Since $\mathrm{ST}^2_x(\psi) \in \mathbf{FO}^2$, ψ^+ is a formula of $\mathbf{IF}^2$. Any independence indication appearing in ψ^+ must be in a context of the form $(\exists\alpha/\beta)$, where β is bound by a universal quantifier $\forall\beta$. Let, then, $\psi^\times$ be the result of having turned all those white diamonds $\Diamond$ in ψ black,

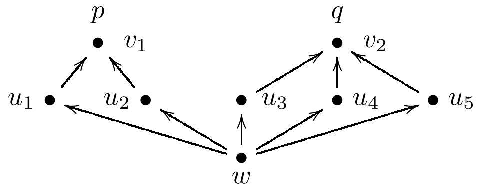


FIGURE 3.

that correspond to an existential quantifier in $\mathrm{ST}_x^2(\psi)$ which has become slashed via the IF procedure leading from ψ to ψ^+; and having also turned all those white boxes $\Box$ in ψ black, that correspond to a universal quantifier in $\mathrm{ST}_x^2(\psi)$ binding the slashed variable of some existential slashed quantifier in ψ^+. It is easy to see that $\mathrm{ST}_x^{\mathbf{IF}}(\psi^\times)$ is, by syntactic criteria, identical to the formula ψ^+. Q.E.D.

4.5 Expressive power

The expressivity and decidability properties of the logic $\mathbf{IFML_{SD}}$ are extensively studied in [Tu$_1$Se$_2$06]. Without entering into details concerning the expressive power of $\mathbf{IFML_{SD}}$, let us take an example.

Example 4.3. Consider evaluating the closed formula $\varphi := \blacksquare(\blacklozenge p \vee \blacklozenge q)$ at the root w of the modal structure $\mathfrak{M}$ depicted in Figure 3. Let I_0 be any token valuation and $\bar{\imath}$ any binary string. We claim that $\mathfrak{M}, I_0, \bar{\imath}, w \models \varphi$. To see this, choose I so that $I(\bar{\imath}0) = v_1$ and $I(\bar{\imath}1) = v_2$. (Choosing a valuation I corresponds to picking out, as it were beforehand, states interpreting the two black diamonds $\blacklozenge$ that can come across later in the evaluation.) It suffices to check that $\mathfrak{M}, I, \bar{\imath}, w \models \Box(\blacklozenge p \ \vee \ \blacklozenge q)$.

For this to hold, it must be possible to partition the set $\{u_1, \dots, u_5\}$ into two cells (corresponding to the choice *left* or *right* for the disjunction symbol), so that if u_j belongs to one of the cells, then $\mathfrak{M}, I, \bar{\imath}0, u_j \models \blacklozenge p$; and if u_j belongs to the other cell, then $\mathfrak{M}, I, \bar{\imath}1, u_j \models \blacklozenge q$. Let the cells be $\{u_1, u_2\}$ for *left*, and $\{u_3, u_4, u_5\}$ for *right.* Then the above conditions hold indeed: the former, since $I(\bar{\imath}0)$ is accessible from all states in the former cell, and p is true at $I(\bar{\imath}0)$; and the latter because $I(\bar{\imath}1)$ is accessible from all states in the latter cell, and q is true at $I(\bar{\imath}1)$. The above reasoning shows, then, that $\mathfrak{M}, w \models \varphi$. Observe that the formula φ can be written in the syntax of **IFML** as $\Box((\Diamond/1)p \vee (\Diamond/1)q)$.

For a further example of what can be expressed in terms of $\mathbf{IFML_{SD}}$, let $n \geq 2$ be arbitrary, and think of the formula $\varphi_n :=$

$$\blacksquare(\underbrace{\blacklozenge\top \vee \ldots \vee \blacklozenge\top}_{n-1 \text{ times}}).$$

Evaluated relative to a modal structure $\mathfrak{M} = (M, R, V)$ at a state w, the formula asserts, in effect, that the set $\{v : R(w, v)\}$ can be partitioned into (at most) $n - 1$ cells in such a way that the elements in each cell have a *common successor* along the relation R. Actually the truth condition of φ can be expressed by the first-order formula $\varphi'_n := \exists z_1 \ldots \exists z_{n-1} \forall y(Rxy \rightarrow (Ryz_1 \vee \ldots \vee Ryz_{n-1}))$. The formula φ'_n is in the $(n+1)$-variable fragment of **FO**. On the other hand, it is not difficult to see (by reference to a pebble game argument[8]) that φ'_n is not equivalent to any formula in the n-variable fragment of **FO**. Hence the greater the number n is, the more variables are needed to translate the formula φ_n into first-order logic. As a consequence, we may infer the following fact:

Fact 4.4. For all $n < \omega$, $\mathbf{IFML_{SD}} \not\leq \mathbf{FO}^n$.

Furthermore, we observe that for all $n \geq 2$, the maximum number of nested modal operators in φ_n is 2. Yet whenever $n' > n$, the formulas φ_n and $\varphi_{n'}$ are not equivalent. So we have:

Fact 4.5. For $\mathbf{IFML_{SD}}$, it is *not* the case that up to logical equivalence, there are only finitely many formulas of a given modal depth.

It may be noted that Facts 4.4 and 4.5 are in a striking contrast to the case of basic modal logic, which is translatable into $\mathbf{FO}^2$, and has the property that the number of pairwise non-equivalent formulas of any given modal depth is finite. (For the latter fact, see, e.g., [BldRVe$_1$01, Proposition 2.29].) These and other unorthodox properties of the modal-like logic $\mathbf{IFML_{SD}}$ might suggest that it should be of a rather marginal interest as a modal logic; even its status as a modal logic might thus be questioned. However, it is proven by the present authors in [Tu$_1$Se$_2$06] that satisfiability and validity problems of $\mathbf{IFML_{SD}}$ are decidable in PSPACE. Hence this expressive logic shares with basic modal logic a good deal of its nice computational properties. So we see that the distribution of 'desirable' and 'undesirable' properties may, in modal-like logics, be rather surprising. Actually, one of the most interesting negative properties of $\mathbf{IFML_{SD}}$ is its non-translatability into the guarded fragment of first-order logic, proven in [Tu$_1$Se$_2$06].

5 Collapse of diversity

Two ways to approach independence friendly modal logic have been discussed: one leading from **IFML** to $\mathbf{EIFML}_k$, proceeding via adding independence indications to modal operators much in the same way as is done

[8] As a reference for the usage of pebble games $G^n_m(\mathcal{M}, \mathbf{a}, \mathcal{N}, \mathbf{b})$ to characterize equivalence of structures up to quantifier rank $\leq m$ relative to $\mathbf{FO}^n$, see, e.g., [EbFl$_2$99, pp. 49-50].

in IF first-order logic—the other way being via adjusting several parameters in such a way that an independence friendly logic gets structurally determined. It is fairly evident that the three logics considered here differ in their expressive power. In fact, we have $\mathbf{IFML_{SD}} < \mathbf{IFML} < \mathbf{EIFML}_k$. (For what is known and what is conjectured about the relations of these logics, see Tables 2 and 3 in Section 6.) By contrast, we now show that in some cases—in fact in cases that are extremely common in modal logical contexts—the expressive powers of these logics coincide.

Let us begin with a couple of definitions. If M is a set and $R \subseteq M^2$ is a binary relation, let us write R^+ for the transitive closure of R, and R^* for the reflexive transitive closure of R. The structure (M, R) is a *tree*, if (i) there is a unique element $r \in M$ such that for all $x \in M$, R^*rx; (ii) every element of M has a unique R-predecessor; and (iii) R is acyclic, i.e., there is no x such that R^+xx. Let us say that a tree is *branching*, if no $x \in M$ has precisely one R-successor (no element has 'out-degree' equal to 1). Hence in a branching tree every element has either no R-successors at all, or has at least two R-successors. A k-ary modal structure $\mathfrak{M} = (M, R_1, \ldots, R_k, V)$ is *(branching and) tree-like*, if the structure $(M, \bigcup_{1\le i\le k} R_i)$ is a (branching) tree. A tree-like k-ary modal structure $\mathfrak{M}$ is *proper*, if for all $x, y \in M$ and all $1 \le i, j \le k$: $[(x, y) \in R_i$ and $(x, y) \in R_j]$ implies $i = j$. That is, in a proper tree-like structure no vertices are connected by more than one relation out of the k available ones. Define $\mathfrak{Tree}_k$ as the class of all proper branching tree-like k-ary modal structures. Note that by virtue of clause (iii) in the definition of tree, all accessibility relations $R_1, \ldots, R_k$ of a structure $\mathfrak{M} \in \mathfrak{Tree}_k$ are irreflexive.

We will prove that all logics **IFML**, $\mathbf{EIFML}_k$ and $\mathbf{IFML_{SD}}$ coincide with basic modal logic (and hence with each other) relative to the class $\mathfrak{Tree}_k$. Consider first $\mathbf{EIFML}_k$. If $\varphi = O_1 \ldots O_n\gamma \in \mathbf{EIFML}_k$, $O_{z+1} = (\Diamond_{j_1\ldots j_y}/i_1, \ldots, i_m)$ and $[x, z] \subseteq \{i_1, \ldots, i_m\}$, we say that the operator O_{z+1} involves *independence of a continuous block of predecessors*. This terminology makes sense: by assumption O_{z+1} is indicated as independent from its immediate predecessor O_z, from the predecessor O_{z-1} of O_z etc., (at least) until O_x. (The smallest number among $i_1, \ldots, i_m$ may well be smaller than x, while its greatest number must be z, since the interval $[x, z]$ is included in $\{i_1, \ldots, i_m\}$.) In what follows, we will rewrite any operator $(\Diamond_{j_1\ldots j_y}/i_1, \ldots, i_{m+m'})$, as given by the syntax, in the form $(\Diamond_{j_1\ldots j_y}/i_1, \ldots, i_m, i'_1, \ldots, i'_{m'})$, where the integers $i_1, \ldots, i_m$ refer by stipulation to modal operators, and the integers $i'_1, \ldots, i'_{m'}$ to propositional connectives.

Lemma 5.1. **(a)** If $\varphi \in \mathbf{EIFML}_k$, let φ^- be the result of replacing all independent diamonds $(\Diamond_{j_1\ldots j_y}/i_1, \ldots, i_m, i'_1, \ldots, i'_{m'})$ in φ by the corresponding diamond $(\Diamond_{j_1\ldots j_y}/i'_1, \ldots, i'_{m'})$ involving no independencies of modal operators. Relative to $\mathfrak{Tree}_k$, φ is equivalent to φ^-. **(b)** If no diamonds in $\varphi \in \mathbf{EIFML}_k$ contain independencies of modal operators, let φ^- be the result of replacing all independent diamonds $(\Diamond_{j_1\ldots j_y}/i'_1,\ldots,i'_{m'})$ in φ by the simple diamond $\Diamond_{j_1\ldots j_y}$. Relative to $\mathfrak{Tree}_k$, φ is equivalent to φ^-.

Proof. **(a)** Let $\mathfrak{M} \in \mathfrak{Tree}_k$ and $w \in M$, and assume that $\mathfrak{M}, w \models \varphi$. Suppose φ contains a diamond $O_{z+1} = (\Diamond_{j_1\ldots j_y}/i_1,\ldots,i_m,i'_1,\ldots,i'_{m'})$ involving independence of a continuous block of predecessors $O_x,\ldots,O_z$ such that $O_x = \Box_{j_1\ldots j_x}$. Suppose a play proceeds to evaluating the operator O_{z+1}. Then the strategy function σ_{z+1} corresponding to O_{z+1} given by $\exists$'s winning strategy (which exists by assumption) satisfies: $\sigma_{z+1}(p_0,p_1,\ldots,p_z) = \sigma_{z+1}(p_0,p'_1\ldots,p'_z)$, where the sequence of choices from the domain associated with p_x is $\varsigma = (w,w_1,\ldots,w_r)$ and the one associated with p'_x is $\varsigma' = (w,w'_1,\ldots,w'_r)$, and $w_r \neq w'_r$. (This is because $\mathfrak{M} \in \mathfrak{Tree}_k$ is branching and so in the two plays $\forall$ has chosen pairwise incomparable and hence distinct states when choosing for the box O_x.) But this is impossible, since in a tree no distinct nodes can have a common successor and so σ_{z+1} cannot be a strategy function involved in a winning strategy.

If the longest possible continuous block of predecessors of O_{z+1} contains only diamonds, then O_{z+1} may trivially be replaced by $(\Diamond_{j_1\ldots j_y}/i'_1,\ldots,i'_{m'})$.

Finally, if φ contains no operator involving independence of a continuous block of predecessors, then all operators $(\Diamond_{j_1\ldots j_y}/i_1,\ldots,i_m,i'_1,\ldots,i_{m'})$ in φ satisfy: either the list $i_1,\ldots,i_m$ is empty, or subordinate to the closest box (if any) identified by an integer in the list, there is a modal operator superordinate to the diamond and not identified by any integer in the list. Hence, if w_r is the most recent choice from the domain made before arriving at the position where $\exists$ must make a choice for the diamond $(\Diamond_{j_1\ldots j_y}/i_1,\ldots,i_m,i'_1,\ldots,i_{m'})$, then, to put it intuitively, $\exists$'s move for the diamond is allowed to depend on w_r. But there is a uniquely determined path in the tree-like structure $\mathfrak{M}$ leading from w to w_r, whence $\exists$ can infer all previous choices made in the relevant partial play. Hence the diamond $(\Diamond_{j_1\ldots j_y}/i_1,\ldots,i_m,i'_1,\ldots,i_{m'})$ may, without changing the truth condition, be replaced by $(\Diamond_{j_1\ldots j_y}/i'_1,\ldots,i'_{m'})$.

(b) Let μ be the distribution of modality types associated with φ. Suppose that $\mathfrak{M}, w \models \varphi$. Consider a diamond $(\Diamond_{j_1\ldots j_y}/i'_1,\ldots,i'_{m'})$ appearing in φ. (If none exists, there is nothing to prove.) If the indicated independencies from conjunctions correspond, as determined by μ, to the requirement of reaching one state along several accessibility relations, then $\exists$'s winning strategy in $G(\varphi,\mathfrak{M},w)$ will choose such a state. But this is impossible, because $\mathfrak{M}$ is a proper tree-like structure and hence no such state exists. On the other hand, if the indicated independencies from conjunctions correspond to making a choice along one and the same accessibility relation irrespective of what the choices for those conjunctions were, then the independent diamond can be replaced by the simple diamond. Finally, if in the diamond considered there are only independencies from disjunctions, the formula says the same as the result of replacing the independent diamond with a simple diamond. Q.E.D.

Recall that $\mathbf{ML}_k$ stands for the polymodal basic modal logic, evaluated relative to k-ary modal structures. We are in a position to prove:

Theorem 5.2. (a) For all $k \geq 1$, $\mathbf{EIFML}_k$ coincides with $\mathbf{ML}_k$ over $\mathfrak{Tree}_k$. **(b)** Both **IFML** and $\mathbf{IFML}_{\mathbf{SD}}$ coincide with **ML** over $\mathfrak{Tree}_1$.

Proof. Statement (a) follows by Lemma 5.1; and statement (b) by an argument exactly like the one presented for item (a) in the proof of Lemma 5.1. Q.E.D.

The class of tree-like structures is omnipresent in modal logic. In particular, any **ML**-formula that has a model at all, has a tree-like model. (Cf., e.g., [BldRVe$_1$01, Proposition 2.15].) The class $\mathfrak{Tree}_k$ discerned above is quite representative a subclass of all tree-like models from the viewpoint of basic modal logic. (It is not difficult to see that any satisfiable polymodal formula $\varphi \in \mathbf{ML}_k$ is satisfied in a structure $\mathfrak{M} \in \mathfrak{Tree}_k$.) Hence it is of interest to see that the additional expressive power of the IF modal languages discussed in the present paper does *not* lie in their capacity to distinguish such tree-like models. Relative to $\mathfrak{Tree}_k$ the three logics do not exceed what already their common core, basic modal logic, is able to express.

There is, at least tentatively, a positive methodological side to our negative expressivity result. Namely, one can propose to turn the tables and suggest that a result such as Theorem 5.2 points to a feature that *any* IF version of basic modal logic should exhibit.[9] That is, results of this type can be used in assessing the general question as to the 'nature' of IF modal logics. From this perspective, indeed it seems reasonable to require that IF modal logics of the appropriate kind precisely *should* coincide with basic modal logic on the class of trees discussed; if a logic does not, it cannot be properly called an IF modal logic in the sense intended. This systematic idea alone brings some order in the manifold of different logics that could conceivably be termed IF modal logics. However, it must be noted that deciding the precise characteristics of a family of logics, such as IF modal logics, is bound to leave some room for discussion; the same holds for the acceptance criteria of any exclusive club of logics—modal, first-order, or what not. What is more, when applying the framework introduced in Section 4, we need not choose a fragment of basic modal logic as the class of modal formulas we start with. Choosing for instance basic tense logic, or

[9] We are indebted to the anonymous referee for pointing out this positive side of our negative result.

first-order modal logic, will likewise result in a system that can, in a generic sense, be termed an IF modal logic. And such logics need not satisfy any specific conditions that may be necessary for IF modal logics corresponding to some different choice of input modal formulas; for instance, there is in general no reason why they should meet the conditions that IF modal logics emerging from basic modal logic actually satisfy.

6 Concluding remarks

In this paper we aimed to discuss two different ways of formulating independence friendly modal logic. To achieve this, we began by surveying and further studying IF modal logics of one of these two kinds, i.e., those obtained from basic modal logic by introducing a suitably interpreted slash device to the syntax (Sections 2 and 3). Now one respect in which modal logic differs from first-order logic is that syntactically, modal operators do not carry variables, whereas quantifiers do. When subject to suitable syntactic restrictions, these variables can easily be employed in referring to particular tokens of quantifiers, whereas no similar syntactic mechanism is available in standard modal logic. This is why in the approaches such as those discussed in Sections 2 and 3, one must introduce an identification method by means of which to single out those tokens of modal operators from whose logical (priority) scope one wishes to exempt, say, a given diamond. On conceptual grounds one might find introducing such identification methods into the syntax less than fortunate. One could argue that independence is a relation between (syntactically manifest) variables, and suggest that since modal syntax does not offer any such variables, it does not really make sense to attempt formulating an IF modal logic. According to such a viewpoint, adding for instance indices to modal operators to make reference to tokens of such operators possible, would be an *ad hoc* move from the perspective of what modal logic is about.

The present authors do not share the ideas on which such a critique is based. We hold independence to be a relation between tokens of logical operators, not first and foremost between syntactically manifest variables. However, we hope to have made it clear in Section 4 that even if independence was considered precisely as a relation between variables, an independence friendly modal logic analogous to IF first-order logic can be defined by considering *fragments of IF first-order logic*. This is the framework of the IF modal logics of the second kind considered in the present paper. The particular fragment to which we gave attention, the result of taking the standard translation of **ML** into $\mathbf{FO}^2$, and applying a certain IF procedure to the resulting class of first-order formulas, even turned out to be of further interest. Namely, we found a modal-like logic $\mathbf{IFML_{SD}}$, with a compositional semantics, capturing the relevant subfragment of (the

L	*Expressivity*	*Satisfiability*	*Validity*
$\mathbf{ML}$	$\mathrm{L} < \mathbf{FO}^2$	PSPACE	PSPACE
$\mathbf{IFML_{PR}}$	$\mathbf{ML} < \mathrm{L} < \mathbf{FO}^3$, $\mathbf{IFML_{SD}} \not\leq \mathrm{L} < \mathbf{IFML}$	$\leq$ SUPEREXP	PSPACE
$\mathbf{IFML}$	$\mathrm{L} \not\leq \mathbf{FO}$	?	?
$\mathbf{EIFML}_k$	$\mathbf{IFML} < \mathrm{L} \not\leq \mathbf{FO}$	?	?
$\mathbf{EIFML}_=$	$\mathrm{L} \not\leq \mathbf{FO}$	undecidable	?
$\mathbf{IFML_{SD}}$	$\mathrm{L} \not\leq \mathbf{FO}^n$, $\mathrm{L} < \mathbf{FO}$, $\mathrm{L} < \mathbf{IFML}$	PSPACE	PSPACE

Table 2. Known results on (IF) modal logics.

L	*Expressivity*	*Validity*
$\mathbf{IFML}$		PSPACE
$\mathbf{EIFML}_k$		PSPACE
$\mathbf{EIFML}_=$		PSPACE
$\mathbf{IFML_{SD}}$	$\mathbf{IFML_{PR}} \not\leq \mathrm{L}$	

Table 3. Conjectures on IF modal logics.

2-variable fragment of) IF first-order logic.

Table 2 lists the main results known about the various logics discussed in the present paper. $\mathbf{EIFML}_=$ stands for the polymodal $\mathbf{EIFML}_k$, one of whose accessibility relations is rigidly interpreted as equality. In Table 3, some conjectures about the various IF logics are presented. The conjecture to the effect that $\mathbf{IFML_{PR}}$ cannot be translated into $\mathbf{IFML_{SD}}$ holds fairly obviously, but the requisite tool called for by the standard proof technique (viz. an appropriate bisimulation relation) has not as yet been formulated in the literature.

Although we feel that the logics discussed and studied in the present paper are interesting in their own right, we think that more generally, they help to see the interest of the grand program of independence-friendliness in logic—that is, repairing Frege's fallacy also outside of first-order logic.

References

[Ba$_2$+03] P. Balbiani, N.-Y. Suzuki, F. Wolter & M. Zakharyaschev, eds. *Advances in Modal Logic 4*. King's College Publications, 2003.

[BldRVe$_1$01] P. Blackburn, M. de Rijke & Y. Venema. *Modal Logic*, *Cambridge Tracts in Theoretical Computer Science* 53. Cambridge University Press, 2001.

[Br$_0$00] J.C. Bradfield. Independence: Logics and concurrency. In [Cl$_1$Sc$_2$00, pp. 247–261].

[Br$_0$Fr$_4$02a] J.C. Bradfield & S.B. Fröschle. Independence-friendly modal logic and true concurrency. *Nordic Journal of Computing* 9(2):102–117, 2002.

[Ca$_1$Ho$_1$01] P.J. Cameron & W. Hodges. Some combinatorics of imperfect information. *Journal of Symbolic Logic* 66(2):673–684, 2001.

[Cl$_1$Sc$_2$00] P. Clote & H. Schwichtenberg, eds. *Computer Science Logic, 14th Annual Conference of the EACSL, Fischbachau, Germany, August 21–26, 2000, Proceedings*, *Lecture Notes in Computer Science* 1862. Springer, 2000.

[EbFl$_2$99] H.-D. Ebbinghaus & J. Flum. *Finite Model Theory.* Springer, 1999.

[Fe$_2$Fr$_3$Hi$_0$89] J.E. Fenstad, I.T. Frolov & R. Hilpinen, eds. *Logic, Methodology and Philosophy of Science VIII*, *Studies in Logic and the Foundations of Mathematics* 126. Elsevier, 1989.

[Ga$_2$St$_0$Wa$_4$95] K. Gavroglu, J. Stachel & M. Wartofsky, eds. *Physics, Philosophy and the Scientific Community.* Kluwer Academic Publishers, 1995.

[Go$_4$Ho$_2$Ve$_1$06] G. Governatori, I. Hodkinson & Y. Venema, eds. *Advances in Modal Logic 6.* College Publications, 2006.

[Hi$_1$73] J. Hintikka. *Logic, Language-Games and Information: Kantian Themes in the Philosophy of Logic.* Clarendon Press, 1973.

[Hi$_1$95] J. Hintikka. What is elementary logic? In [Ga$_2$St$_0$Wa$_4$95, pp. 301–326].

[Hi$_1$96] J. Hintikka. *The Principles of Mathematics Revisited.* Cambridge University Press, 1996.

[Hi$_1$97] J. Hintikka. No scope for scope? *Linguistics and Philosophy* 20(5):515–544, 1997.

[$\mathrm{Hi_1Sa_489}$] J. Hintikka & G. Sandu. Informational independence as a semantical phenomenon. In [$\mathrm{Fe_2Fr_3Hi_089}$, pp. 571–589].

[$\mathrm{Ho_197a}$] W. Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL* 5(4):539–563, 1997.

[$\mathrm{Ho_197b}$] W. Hodges. Some strange quantifiers. In [$\mathrm{MyRo_4Sa_197}$, pp. 51–65].

[$\mathrm{Hy_1Tu_105}$] T. Hyttinen & T. Tulenheimo. Decidability of IF modal logic of perfect recall. In [$\mathrm{Sc_1}$+05, pp. 111–131].

[$\mathrm{MyRo_4Sa_197}$] J. Mycielski, G. Rozenberg & A. Salomaa, eds. *Structures in Logic and Computer Science, A Selection of Essays in Honor of Andrzej Ehrenfeucht*, *Lecture Notes in Computer Science* 1261. Springer, 1997.

[$\mathrm{Sa_493}$] G. Sandu. On the logic of informational independence and its applications. *Journal of Philosophical Logic* 22(1):29–60, 1993.

[$\mathrm{Sc_1}$+05] R. Schmidt, I. Pratt-Hartmann, M. Reynolds & H. Wansing, eds. *Advances in Modal Logic 5*. King's College London Publications, 2005.

[$\mathrm{Se_206}$] M. Sevenster. *Branches of imperfect information: logic, games, and computation*. Ph.D. thesis, Universiteit van Amsterdam, 2006. *ILLC Publications* DS-2006-06.

[$\mathrm{Tu_103}$] T. Tulenheimo. On IF modal logic and its expressive power. In [$\mathrm{Ba_2}$+03, pp. 475–498].

[$\mathrm{Tu_104}$] T. Tulenheimo. *Independence Friendly Modal Logic*. Ph.D. thesis, Department of Philosophy, University of Helsinki, 2004.

[$\mathrm{Tu_1Se_206}$] T. Tulenheimo & M. Sevenster. On modal logic, IF logic and IF modal logic. In [$\mathrm{Go_4Ho_2Ve_106}$, pp. 481–501].

[Vä07] J. Väänänen. *Dependence Logic: A New Approach to Independence Friendly Logic*, *London Mathematical Society Student Texts* 70. Cambridge University Press, 2007.

[vB76] J.F.A.K. van Benthem. *Modal Correspondence Theory*. Ph.D. thesis, Mathematisch Instituut & Instituut voor Grondlagenonderzoek, Universiteit van Amsterdam, 1976.

Team Logic*

Jouko Väänänen

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam, The Netherlands

Matematiikan ja tilastotieteen laitos
Helsingin yliopisto
P.O. Box 68
00014 Helsinki, Finland

vaananen@science.uva.nl

Abstract

Team logic is the logic of functional dependencies. We define the basic logical operations of team logic, establish its relationship with independence friendly and second order logic, and give it a game theoretic characterization.

1 Introduction

Let a vocabulary[1] L and an L-structure $\mathcal{M}$ with universe M be given. In this paper we study *functional dependencies* in $\mathcal{M}$. This is in contrast to the traditional approach in logic of studying *relational dependencies* in $\mathcal{M}$. Our atomic dependence relations state the existence of a functional dependence without giving any definition for the function that carries the dependence. This gives the whole topic a second order flavor. It seems to the author that although functional dependence in databases has been studied (starting with [Ar74]), a general theory of more complex types of dependence is new. (For a more detailed recent study, see [Vä07].) Our theory is based on [Ho$_1$97b] and [VäHo$_1\infty$].

If L has an n-ary function symbol f, there is immediately an apparent dependence relation in $\mathcal{M}$, namely the dependence of each $a = f^{\mathcal{M}}(a_1, \ldots, a_n)$ on $a_1, \ldots, a_n$. If the function $f^{\mathcal{M}}$ is constant, the dependence is of a singular kind, not commonly called dependence on $a_1, \ldots, a_n$ at all. On the

* This paper was written while the author was a guest in the Newton Institute (Cambridge, U.K.) as part of the special program *Logic and Computation.* The author is grateful to the Newton Institute for its support. Research was also partially supported by grant 40734 of the Academy of Finland. I am indebted to the referee for pointing out some inaccuracies and suggesting improvements, and to Fred Oakley for permission to use the photo in Figure 2.

[1] The vocabulary may contain constant, relation and function symbols.

other hand, if the function is one to one, we have a dependence that is so perfect that it can even be reversed. In this case dependence is carried by a function that has a name in the vocabulary. In general we have completely abstract functions carrying the dependence with no name in the vocabulary and even no definition in the language, what so ever.

If L has an n-ary relation symbol R, there is a relational dependence relation in $\mathcal{M}$, namely the mutual dependence of $a_1, \ldots, a_n$ such that $(a_1, \ldots, a_n) \in R^{\mathcal{M}}$ on each other. However, even if such a relation between the elements $a_1, \ldots, a_n$ binds the elements together in an obvious sense, it need not in general be a (functional) dependence in the sense of the current approach.

Team logic is the logic of (functional) dependence. We define the basic concepts of team logic and use it to analyze dependence. The negation-free part of team logic turns out to correspond naturally to independence friendly logic and thereby to the existential part of second order logic. Team logic itself can be seen as being the natural closure of independence friendly logic under (classical) negation. In expressive power team logic is in a natural sense equivalent to full second order logic. We give a game-theoretic characterization of team logic by means of an appropriate Ehrenfeucht-Fraïssé game.

2 Agents and teams

An *agent* is any finite mapping s from a domain $\mathrm{dom}(s)$ into M. Elements of $\mathrm{dom}(s)$ are called *features* or *fields*, sometimes *attributes*, depending on the application[2]. Unless stated otherwise, to be specific, the domain of an agent is a finite set of natural numbers. The *modification* $s(a/n)$ of an agent s maps n to a but agrees otherwise with s. The *value* of a term t, built up from variables x_n and symbols of L, on an agent s is the element $t\langle s\rangle$ of M defined in the usual way by $c\langle s\rangle = c^{\mathcal{M}}$, $x_n\langle s\rangle = s(n)$, and $ft_1 \ldots t_n\langle s\rangle = f^{\mathcal{M}}(t_1\langle s\rangle, \ldots, t_n\langle s\rangle)$. For this to make sense the domain of s has to contain n whenever x_n occurs in t.

Building on [Ho$_1$97a] and [Ho$_1$97b] we take the position that dependence, just like any other pattern, cannot be manifested by one event, observation or agent (which is our terminology) but needs a series or a group of events, observations or agents. We call these series or groups manifesting dependence teams.

A *team* (see Figure 1) is any set X of agents with the same domain $\mathrm{dom}(X)$. Table 1 presents a generic team in the form of a table. Here are some examples of teams:

[2] Agents are also called assignments or tuples. If we think of an agent s as an assignment, the elements of the domain are variables x_n or their indexes n, depending on how values of terms are defined.

	Features			
Agent	m_1	m_2	$\ldots$	m_n
s_1	$s_1(m_1)$	$s_1(m_2)$	$\ldots$	$s_1(m_n)$
s_2	$s_2(m_1)$	$s_2(m_2)$	$\ldots$	$s_2(m_n)$
s_3	$s_3(m_1)$	$s_3(m_2)$	$\ldots$	$s_3(m_n)$
s_4	$s_4(m_1)$	$s_4(m_2)$	$\ldots$	$s_4(m_n)$
s_5	$s_5(m_1)$	$s_5(m_2)$	$\ldots$	$s_5(m_n)$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$

TABLE 1. A generic team as a table.

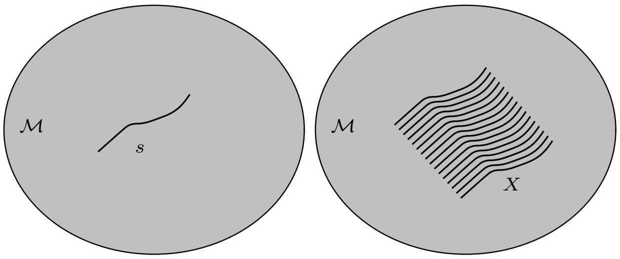


FIGURE 1. A model with an assignment and a model with a team.

Team 1 A team of human genomes (Figure 2). Here the agents are individual genomes of individual people. The fields (or features) are the individual genes that are observed. Potentially there are tens of thousands of possible fields to consider, and billions of agents. If we add to such a team new fields related to medical data of the person in question, we get a team the dependencies of which may give crucial data about the role of hereditary factors in medical science. For example, we may ask:

1. Does a certain gene or gene combination (significantly) *determine* a given hereditary disease in the sense that a patient with (a fault in) those genes has a high risk of the disease?
2. Is a disease *totally dependent* on a gene in the sense that every gene combination that (significantly) determines the disease contains that particular gene.
3. Is a gene (merely) *dependent* on a gene in the sense that the

disease is (significantly) determined by some gene combination with the gene but not without.

4. Is a disease *totally independent* of a gene in the sense that no gene combination that (significantly) determines the disease contains that particular gene.
5. Is a gene (merely) *independent* of a gene in the sense that some gene combinations (significantly) determine the disease without containing that particular gene.

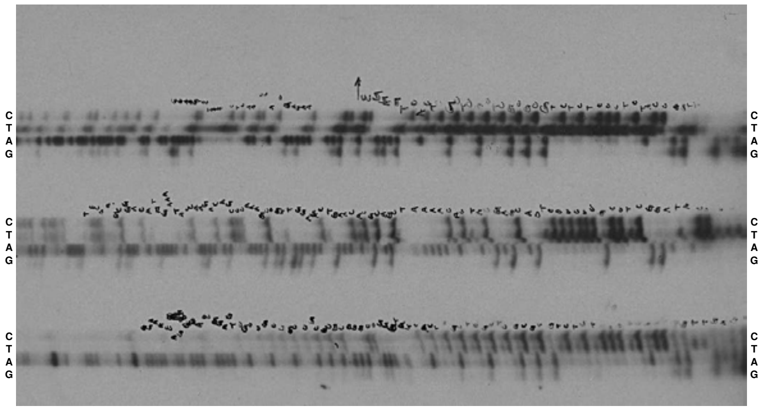


FIGURE 2. A piece of a team of genomes.

Team 2 Game history team: Imagine a game, any game. Game moves are the features of this team, plays are the agents, and a collection of plays is a team. Thus a player is identified with his or her behavior in the game. It may be relevant to know answers to the following kinds of questions:

1. What is the strategy that a player (e.g. "Nature") is following, or is he or she following any strategy at all?
2. Is a player using information about his or her (or other players') moves that he or she is committed not to use?

Team 3 Every formula $\varphi(x_1, \ldots, x_n)$ of any logic and structure $\mathcal{M}$ give rise to the team of all assignments that satisfy $\varphi(x_1, \ldots, x_n)$ in $\mathcal{M}$. This is a definable team and perhaps the most obvious team arising in logic. It may manifest functional dependencies on the structure, and depending on the logic, these dependencies can or cannot be expresses in the logic.

Team 4 Every first order sentence φ and structure $\mathcal{M}$ give rise to teams consisting of assignments that arise in the semantic game of φ and $\mathcal{M}$. If $\mathcal{M} \models \varphi$ and the winning strategy of II is τ, a particularly coherent team consists of all plays of the semantic game in which II uses τ. The same holds of independence friendly logic of [Hi$_1$96].

Team 5 A team of robots building a car. Each agent s, i.e. robot, has features encoded by the function s. For example, if the domain of the agents is

$$\{\text{reach}, \text{payload}, \text{joint range}, \text{speed}, \text{weight}, \text{paints}, \text{welds}\},$$

we can have a team as in Table 2. Here weight depends on the payload

robot	`reach` (mm)	`payload` (kg)	`range` (°)	`speed` (°/sec)	`weight` (kg)	`paints`	`welds`
1	653	2.5	170	328	28	true	false
2	653	2.5	170	328	28	true	false
3	653	2.5	190	328	28	true	false
4	653	5	170	328	35	true	false
5	653	5	170	300	35	false	true
6	653	5	170	300	35	false	true
7	653	5	360	328	35	false	true

TABLE 2. A team of robots

but not on speed as robots 3 and 4 have the same speed but different weight. Note that the team can be divided into two subteams one of which paints and the other does welding. This team raises the issue whether a team can have two agents with exactly the same values on all features. According to our definition the two agents are in such a case the same. So the above table describing the team is misleading. Robots 1 and 2 are the same, as are robots 5 and 6. If we added the robot number to the domain, the robots would all be different agents. This indicates an extensionality phenomenon in our approach.

3 Dependence types

Our starting point was the idea that teams can manifest dependencies. The different ways that this happens give rise to the concept of a dependence type[3] that we now define. Some dependence types are of an atomic nature in that they are not decomposed further into smaller dependence types. The atomic types are

[3] Types correspond to formulas. We use the word 'type' to emphasize the difference between truth and dependence.

- $t = t'$: Every agent $s \in X$ satisfies $t\langle s\rangle = t'\langle s\rangle$. The team of Table 3 is of type $x_3 = x_4$.
- $\neg t = t'$: Every agent $s \in X$ satisfies $t\langle s\rangle \neq t'\langle s\rangle$. The team of Table 3 is of type $\neg x_2 = x_4$.
- $Rt_1 \ldots t_n$: Every agent $s \in X$ satisfies $(t_1\langle s\rangle, \ldots, t_n\langle s\rangle) \in R^{\mathcal{M}}$. The team of Table 3 is of type $x_2 < x_4$.
- $\neg Rt_1 \ldots t_n$: Every agent $s \in X$ satisfies $(t_1\langle s\rangle, \ldots, t_n\langle s\rangle) \notin R^{\mathcal{M}}$. The team of Table 3 is of type $\neg x_4 < x_2$.
- $=(t_1, \ldots, t_n)$: Every two agents $s, s' \in X$ that satisfy

$$\begin{aligned} t_1\langle s\rangle &= t_1\langle s'\rangle \\ &\vdots \\ t_{n-1}\langle s\rangle &= t_{n-1}\langle s'\rangle, \end{aligned}$$

  also satisfy $t_n\langle s\rangle = t_n\langle s'\rangle$. This type is called the *dependence* type. The team of Table 3 is of type $=(x_1, x_2)$ but not of type $=(x_2, x_1)$. There are two special cases. The first is $=(t)$. This is the type of teams in which the value of t is constant. Team 5 above is of type $=$(`reach`). The second special case is the type $=()$. This is the type of all teams what so ever and is given a special symbol $\top$.

- $\neg =(t_1, \ldots, t_n)$: This is the type of the empty team $\varnothing$. This should be distinguished from the type of an "impossible" team, that is, the type which is not the type of any team what so ever. We shall return to this type later. The reason for allowing only the empty team to be of the type $\neg =(t_1, \ldots, t_n)$ is the following: We want types to be closed downwards. If a non-empty team was of type $\neg =(t_1, \ldots, t_n)$, there would have to be a singleton team $\{s\}$ of the same type. But singleton teams are always of type $=(t_1, \ldots, t_n)$.

The types $t = t'$, $\neg t = t'$, $Rt_1 \ldots t_n$ and $\neg Rt_1 \ldots t_n$ occur in [Ho$_1$97a] and [Ho$_1$97b]. The dependence type $=(t_1, \ldots, t_n)$ is introduced in [VäHo$_1\infty$]. The empty team is of every atomic type, in particular both of type $t = t'$ and $\neg t = t'$. From atomic dependence types we can build more complex ones with *team operations*, which are the following:

- Negation $\sim\varphi$: The team is not of type φ. The team of Table 3 is of type $\sim =(x_2, x_1)$. Note that $\sim Rt_1 \ldots t_n$ is quite different from $\neg Rt_1 \ldots t_n$. The former says *some* agent s in the team fails to satisfy $(t_1\langle s\rangle, \ldots, t_n\langle s\rangle) \in R^{\mathcal{M}}$, while the latter says *all* fail. Even more

	Domain			
Agent	1	2	3	4
s_1	100	9	10	10
s_2	20	9	10	10
s_3	21	1	10	10
s_4	1	2	11	11
s_5	101	1	11	11

TABLE 3. A team in $(\mathbb{N}, <)$.

	Domain			
Agent	1	2	3	4
s_1	100	9	10	10
s_2	20	9	10	10
s_5	101	1	11	11

	Domain			
Agent	1	2	3	4
s_3	21	1	10	10
s_4	1	2	11	11

TABLE 4. Two teams in $(\mathbb{N}, <)$.

dramatic is the difference between $\sim\!=\!(t_1, \ldots, t_n)$ and $\neg\!=\!(t_1, \ldots, t_n)$. The former says some s and s' in the team give the same value to $(t_1, \ldots, t_{n-1})$ but a different value to t_n, while the latter says there are no agents in the team at all. Finally $\sim\!=\!()$ is the type of no team. Thus we have two negations, $\neg$ and $\sim$, but the former can be applied to atomic types only. Its range of applicability can be extended to what we call $\sim$-free part of team logic.

- Conjunction $\varphi \wedge \psi$: The team is both of type φ and of type ψ. The team of Table 3 is of type $x_3 = x_4 \wedge \neg x_2 = x_4$.

- Tensor $\varphi \otimes \psi$: The team is the union of a team of type φ and a team of type ψ. Team 5 is of type `paints`$\otimes$`welds`. This indicates the role of $\otimes$ in expressing co-operation skills of teams. The team of Table 3 is of type $=\!(x_3, x_2) \otimes =\!(x_3, x_2)$, as Table 4 shows. Note that that team is not of type $=\!(x_3, x_2)$. So tensor is not idempotent as conjunction is.

- Existential quantifier $\exists x_n \varphi$: The agents $s \in X$ can be modified to $s(a_s/n)$ in such a way that the team $\{s(a_s/n); s \in X\}$ is of type

φ. The team of Table 3 is of type $\exists x_1(x_1 < x_2)$ and also of type $\exists x_5(x_5 < x_2)$.

- Shriek quantifier $!x_n\varphi$: A team X is of this type if the team of all agents $s(a/n)$, where $a \in M$ and $s \in X$, is of type φ. The team of Table 3 is of type $!\,x_1(x_1 < x_3 \otimes x_2 < x_1)$. but not of type $!\,x_1(x_2 < x_1)$.

More formally:

Definition 3.1. Let L be a vocabulary. The set of *types* is defined as follows: If $t, t', t_1, \ldots, t_n$ are terms of the vocabulary L, and R is a relation symbol in L, then the following are types

$$\begin{array}{l} t = t'. \\ \neg t = t'. \\ Rt_1 \ldots t_n. \\ \neg Rt_1 \ldots t_n. \\ =(t_1, \ldots, t_n). \\ \neg =(t_1, \ldots, t_n). \end{array}$$

If φ and ψ are types, then so are:

$$\begin{array}{l} \sim \varphi. \\ \varphi \wedge \psi. \\ \varphi \otimes \psi. \\ \exists x_n \varphi. \\ !x_n \varphi. \end{array}$$

The concept "team X is of type φ" is defined as follows:

- X is of type $t = t'$ iff for all $s \in X(t\langle s\rangle = t'\langle s\rangle)$.
- X is of type $\neg t = t'$ iff for all $s \in X(t\langle s\rangle \neq t'\langle s\rangle)$.
- X is of type $Rt_1 \ldots t_n$ iff for all $s \in X(t_1\langle s\rangle, \ldots, t_n\langle s\rangle) \in R^{\mathcal{M}}$.
- X is of type $\neg Rt_1 \ldots t_n$ iff for all $\in X(t_1\langle s\rangle, \ldots, t_n\langle s\rangle) \notin R^{\mathcal{M}}$.
- X is of type $=(t_1, \ldots, t_n)$ iff for all $s, s' \in X([t_1\langle s\rangle = t_1\langle s'\rangle \& \ldots \& t_{n-1}\langle s\rangle = t_{n-1}\langle s'\rangle]$ implies $t_n\langle s\rangle = t_n\langle s'\rangle)$.
- X is of type $\neg =(t_1, \ldots, t_n)$ iff $X = \varnothing$.
- X is of type $\sim \varphi$ iff X is not of type φ.
- X is of type $\varphi \wedge \psi$ iff X is both of type φ and of type ψ.

- X is of type $\varphi \otimes \psi$ iff $X = Y \cup Z$, where Y is of type φ and Z is of type ψ.
- X is of type $\exists x_n \varphi$ iff there is a function $s \mapsto a_s$ from X to M so that the team $\{s(a_s/n); s \in X\}$ is of type φ.
- X is of type $!x_n\varphi$ iff the team $\{s(a/n) : a \in M, s \in X\}$ is of type φ.

We can define further types from the above ones:

- Disjunction $\varphi \vee \psi$: This is the type $\sim(\sim\varphi \wedge \sim\psi)$, i.e. the type of teams that are of type φ or of type ψ (or both).
- Universal quantifier $\forall x_n\varphi$: This is the type $\sim\exists x_n \sim\varphi$, i.e. the type of teams X such that whenever the agents $s \in X$ are modified to some $s(a_s/n)$, then the new team $\{s(a_s/n); s \in X\}$ is of type φ.
- Implication $\varphi \rightarrow \psi$: This is the type $\sim\varphi \vee \psi$.
- Lollipop $\varphi \multimap \psi$: This is the type $\sim(\varphi \otimes \sim\psi)$, i.e. if whenever it is represented as the union of two teams the first of which is of type φ, then the other one is of type ψ.

There are exactly two teams with empty domain, namely $\varnothing$ and $\{\varnothing\}$. From these we get exactly four different types of teams with empty domain:

Symbol	Type	Teams
$\top$	$=()$	$\varnothing, \{\varnothing\}$
$\bot$	$\sim=()$	
1	$\sim\neg=()$	$\{\varnothing\}$
0	$\neg=()$	$\varnothing$

These *dependence values* form the following diamond:

$$\begin{tikzcd}
 & \top & \\
\mathbf{1} \arrow[ur] & & \mathbf{0} \arrow[ul] \\
 & \bot \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Definition 3.2. Suppose φ is a type of a team with empty domain.[4] We define $\mathcal{M} \models \varphi$ to mean that the team $\{\emptyset\}$ is of type φ in $\mathcal{M}$. We then say $\mathcal{M}$ is *of type* φ.

In a sense, team logic is a four-valued logic. The dependence values of types $\varphi \otimes \psi$, $\varphi \wedge \psi$, and $\sim \varphi$ for types φ and ψ of teams with empty domain can be readily given in terms of truth-tables:

$\otimes$	$\top$	$\bot$	**1**	**0**
$\top$	$\top$	$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
1	$\top$	$\bot$	**1**	**1**
0	$\top$	$\bot$	**1**	**0**

$\wedge$	$\top$	$\bot$	**1**	**0**
$\top$	$\top$	$\bot$	**1**	**1**
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
1	**1**	$\bot$	**1**	**0**
0	**0**	$\bot$	**0**	**0**

	$\sim$
$\top$	$\bot$
$\bot$	$\top$
1	**0**
0	**1**

Here are some examples of types: The type

$$=(x_0, x_1) \otimes =(x_0, x_1)$$

is the type of teams in which field 1 depends on field 0 in the weaker sense than functional dependence that the values of field 0 determine at most two values one of which is the value in field 1. Table 5 is an example of

Rank	Salary
A	2000
A	2100
B	2150
B	2220
C	2340
C	2440
D	2500
D	3100
E	3200
E	3710

TABLE 5. Does salary depend on rank?

a team which is of type $=(\texttt{Rank}, \texttt{Salary}) \otimes =(\texttt{Rank}, \texttt{Salary})$. If we denote $=(x_0, x_1)$ by φ, we get weaker and weaker forms of functional dependence by considering the types

$$\begin{array}{l} \varphi \otimes \varphi \\ \varphi \otimes \varphi \otimes \varphi \\ \varphi \otimes \varphi \otimes \varphi \otimes \varphi \\ \ldots \end{array}$$

[4] That is, φ is a 'sentence', it has no free variables.

The type

$$=\!(x_0, x_1) \wedge =\!(x_1, x_0)$$

is the type of teams in which field 0 depends functionally on field 1 and conversely field 1 depends functionally on field 0. We could say that fields 0 and 1 are **mutually functionally dependent**. The type

$$\neg x_0 = x_1 \otimes x_2 = x_3$$

is the type of teams in which each agent s which satisfies $s(0) = s(1)$ also satisfies $s(2) = s(3)$. The type

$$\neg x_0 = x_1 \otimes x_2 = x_3$$

is the type of teams in which each agent s which satisfies $s(0) = s(1)$ also satisfies $s(2) = s(3)$. The type

$$(\neg x_0 = x_1 \otimes x_2 = x_3) \wedge (\neg x_2 = x_3 \otimes x_0 = x_1)$$

is the type of teams in which each agent s which satisfies $s(0) = s(1)$ also satisfies $s(2) = s(3)$ and vice versa. The type

$$\begin{aligned} !\, x_0 \exists x_1 \,!\, x_2 \exists x_3 (=\!(x_2, x_3) \quad & \wedge \quad \neg(x_0 = x_1) \\ & \wedge \quad (\neg x_0 = x_2 \otimes x_1 = x_3) \\ & \wedge \quad (\neg x_1 = x_2 \otimes x_3 = x_0)) \end{aligned}$$

is the type of teams which are non-empty if and only the underlying set M is either infinite or finite and of even cardinality. The type

$$\begin{aligned} !\, x_0 \exists x_1 \,!\, x_2 \exists x_3 (=\!(x_2, x_3) \quad \wedge \quad & (\neg P x_0 \otimes (Q x_1 \wedge \\ & (\neg x_0 = x_2 \otimes x_1 = x_3) \\ & (\neg x_1 = x_3 \otimes x_0 = x_2)))) \end{aligned}$$

is the type of teams which are non-empty if and only if in the underlying structure $\mathcal{M}$ the predicate P and Q satisfy $|P^{\mathcal{M}}| \leq |Q^{\mathcal{M}}|$.

4 Modes of dependence and independence

We shall now use the above concept of dependence type to analyze in more general terms different concepts of dependence and independence.

Let X be a team with domain $\{m_1, \ldots, m_n\}$ in a domain M, as in Table 1. There is an obvious partial order in the powerset of $\{m_1, \ldots, m_n\}$, namely the set-theoretical subset-relation $\subseteq$. We now define a new relation, called the pre-order of functional dependence:

$V \leq W$	V is *functionally dependent* on W, i.e. features in V can be determined if the values of the features in W are known. In symbols, $\forall s, s' \in X((\forall y \in W(s(y) = s'(y)) \to (\forall x \in V(s(x) = s'(x)))))$. Equivalently: V is functionally dependent on $\{w_1, \ldots, w_n\}$, if for all $y \in V$ there is a function f_y such that for all s in X: $s(y) = f_y(s(w_1), \ldots, s(w_n))$.

It is evident from the definition that the pre-order of functional dependence is weaker than the partial order of inclusion in the sense that every subset of a set obviously depends functionally on the set itself. It is more interesting that sometimes a set is functionally dependent on a set disjoint from itself, and a singleton set may be functionally dependent of another singleton set. Some sets may be functionally dependent on the empty set (in that case the feature has to have a constant value). Every set is certainly functionally dependent on the whole universe.

Note that the Armstrong Axioms of functional dependence (see [Ar74]) state exactly the following:

1. $V \leq W$ is a pre-order, i.e. reflexive and transitive.
2. If $V \subseteq W$, then $V \leq W$.
3. If $V \leq W$ and U is arbitrary, then $V \cup U \leq W \cup U$.
4. If $V \leq W$, then there is a minimal $U \subseteq W$ such that $V \leq U$.

These axioms characterize completely when a functional dependence $V \leq W$ follows from given functional dependencies $V_1 \leq W_1, \ldots, V_n \leq W_n$.

Now we can define two versions of dependence, by using the pre-order of determination. Suppose for $W \cap V = \varnothing$. We define:

V is **dependent** on W	There is some minimal $U \geq V$ such that $U \cap V = \varnothing$ and $W \subseteq U$.
V is **totally dependent** on W	For every $U \geq V$ such that $U \cap V = \varnothing$ we have $W \subseteq U$.
V is **independent** of W	There is some $U \geq V$ such that $U \cap V = \varnothing$ and $W \cap U = \varnothing$.
V is **totally independent** of W	For every minimal $U \geq V$ such that $U \cap V = \varnothing$ we have $W \cap U = \varnothing$.
V is **non-determined**	There is no $U \geq V$ such that $U \cap V = \varnothing$. In the opposite case V is called determined.

`star`	`food`	`drink`	`music`
Marlon	pasta	wine	classical
Jack	pasta	beer	classical
Robert	steak	wine	rock
Julia	steak	beer	rock

TABLE 6. Team of favorites.

Note, that it is quite conceivable that two sets V and W of features are *mutually dependent* in the sense that both depend on each other. In the team of Table 6 we can make the following observations: `star` depends on `food`, since $\{\texttt{star}\} \leq \{\texttt{food}, \texttt{drink}\}$. On the other hand, `star` depends also on `music`, since $\{\texttt{star}\} \leq \{\texttt{drink}, \texttt{music}\}$. So `star` is not totally dependent on either `food` or `music` but it is totally dependent on `drink`.

Note that the above concepts are defined with respect to the fields that we have in the domain. Indeed, it seems meaningless to define what independence means in a domain where any *new* fields can be introduced. The new fields can change independence to dependence completely.

Here are some immediate relationships between the introduced concepts of dependence and independence:

1. Every V is totally dependent and totally independent on $\varnothing$.
2. If V is (totally) dependent on W, then V is (totally) dependent on every subset of W.
3. If V is dependent on W, it can still be also independent of W, but not totally, unless $W = \varnothing$.
4. V is independent of $\{x\}$ if and only if V is not totally dependent on $\{x\}$.
5. V is totally independent of $\{x\}$ if and only if V is not dependent on $\{x\}$.

If $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_m\}$, let $=(U, V)$ be the type

$$\bigwedge_{i=1}^{m} =(u_1, \ldots, u_n, v_i).$$

This is the type of teams in which V is functionally dependent on U. It is clear that we can define dependence, independence total dependence, total independence, and non-determinedness in terms of the basic types $=(U, V)$. We suggest that the types of team logic provide a proper framework for an analysis of the variety of different concepts related to dependence.

5 Team algebra

Let D be a fixed domain and $\mathcal{T}$ the set of all teams with domain D. Let us write $\varphi \Leftrightarrow \psi$ if every team of type φ is of type ψ and vice versa. We call this relation *logical equivalence*. The Boolean operations $\wedge, \vee, \sim$ together with $\top$ and $\bot$ obey usual laws of Boolean algebras, and the mapping

$$h(\varphi) = \{X \in \mathcal{T} : X \text{ is of type } \varphi\}.$$

is a homomorphism between dependence types, endowed with the operations $\wedge, \vee$ and $\sim$, and the Boolean algebra of subsets of $\mathcal{T}$:

$$\begin{array}{lcl} h(\varphi \wedge \psi) & = & h(\varphi) \cap h(\psi) \\ h(\varphi \vee \psi) & = & h(\varphi) \cup h(\psi) \\ h(\sim \varphi) & = & \mathcal{T} - h(\varphi) \\ h(\top) & = & \mathcal{T} \\ h(\bot) & = & \varnothing \end{array}$$

The operation $\otimes$ satisfies the laws

$$\begin{array}{lcl} \varphi \otimes \psi & \Leftrightarrow & \psi \otimes \varphi \\ \varphi \otimes (\psi \otimes \theta) & \Leftrightarrow & (\varphi \otimes \psi) \otimes \theta \\ \varphi \otimes (\psi \vee \theta) & \Leftrightarrow & (\varphi \otimes \psi) \vee (\varphi \otimes \theta) \\ \varphi \otimes \bot & \Leftrightarrow & \bot \\ \varphi \otimes \mathbf{0} & \Leftrightarrow & \varphi \\ \mathbf{1} \otimes \mathbf{1} & \Leftrightarrow & \mathbf{1} \\ \top \otimes \top & \Leftrightarrow & \top \end{array}$$

but

$$\begin{array}{lcl} \varphi \otimes (\psi \wedge \theta) & \not\Leftrightarrow & (\varphi \otimes \psi) \wedge (\varphi \otimes \theta) \\ \varphi \wedge (\psi \otimes \theta) & \not\Leftrightarrow & (\varphi \wedge \psi) \otimes (\varphi \wedge \theta) \\ \varphi \vee (\psi \otimes \theta) & \not\Leftrightarrow & (\varphi \vee \psi) \otimes (\varphi \vee \theta) \\ \varphi \otimes \varphi & \not\Leftrightarrow & \varphi \end{array}$$

For quantifiers we have

$$\begin{array}{lcl} \forall x_n(\varphi \wedge \psi) & \Leftrightarrow & \forall x_n\varphi \wedge \forall x_n\psi \\ \forall x_n \forall x_m \varphi & \Leftrightarrow & \forall x_m \forall x_n \varphi \\ \exists x_n(\varphi \vee \psi) & \Leftrightarrow & \exists x_n\varphi \vee \exists x_n\psi \\ \exists x_n \exists x_m \varphi & \Leftrightarrow & \exists x_m \exists x_n \varphi \end{array}$$

The shriek ! commutes with every team operation except $\exists$:

$$\begin{array}{lcl} !x_n \sim \varphi & \Leftrightarrow & \sim !x_n\varphi \\ !\,x_n(\varphi \otimes \psi) & \Leftrightarrow & !\,x_n\varphi \otimes !\,x_n\psi \\ !\,x_n(\varphi \wedge \psi) & \Leftrightarrow & !\,x_n\varphi \wedge !\,x_n\psi \\ !\,x_n(\varphi \vee \psi) & \Leftrightarrow & !\,x_n\varphi \vee !\,x_n\psi \\ !\,x_n\forall x_m\varphi & \Leftrightarrow & \forall x_m\, !\,x_n\varphi \\ !\,x_n\exists x_m\varphi & \not\Leftrightarrow & \exists x_m\, !\,x_n\varphi \end{array}$$

Obviously, there is a lot more one can say about team algebra.

6 Some translations

We show that the $\sim$-free fragment of team logic is equivalent to independence friendly logic. To make our result exact we choose what we believe is the best behaving version of independence friendly logic. This is the *dependence friendly logic* in which the quantifier

$$\exists x_n \backslash t_1 \dots t_m \varphi$$

has the meaning "there is a value for x_n, functionally depending on what the values of $t_1, \dots, t_m$ are, such that φ". In the original independence friendly logic the quantifier

$$\exists x_n / \forall x_1 \dots \forall x_m \varphi$$

had the meaning "there is a value for x_n, independently of what the values of $x_1, \dots, x_m$ are, such that φ". An attempt to understand what "independently" might mean here led the author to the considerations of Section 4 and eventually to the more basic concept of functional dependence.

We can easily define a translation $\varphi \mapsto \varphi^*$ of dependence friendly logic into team logic, but we have to assume that the formula φ of dependence friendly logic is in negation normal form:

$$\begin{array}{lcl}
(t = t')^* & = & t = t' \\
(\neg t = t')^* & = & \neg t = t' \\
(Rt_1 \dots t_n)^* & = & Rt_1 \dots t_n \\
(\neg Rt_1 \dots t_n)^* & = & \neg Rt_1 \dots t_n \\
(\varphi \vee \psi)^* & = & \varphi^* \otimes \psi^* \\
(\varphi \wedge \psi)^* & = & \varphi^* \wedge \psi^* \\
(\exists x_n \backslash t_1, \dots, t_n \varphi)^* & = & \exists x_n (=(t_1, \dots, t_n, x_n) \wedge \varphi^*) \\
(\forall x_n \varphi)^* & = & !x_n \varphi^*
\end{array}$$

It is an immediate consequence of the definitions that for all $\mathcal{M}$, all φ, and all X we have

$$\mathcal{M} \models_X \varphi \text{ in dependence friendly logic if and only if } X \text{ is of type } \varphi^*.$$

So we may consider dependence friendly logic a fragment of team logic, and team logic an extension of dependence friendly logic obtained by adding classical negation.

Conversely, we can define a translation $\varphi \mapsto \varphi^+$ of the $\sim$-free fragment of team logic into dependence friendly logic:

$$\begin{array}{lcl}
(t = t')^+ & = & t = t' \\
(\neg t = t')^+ & = & \neg t = t' \\
(Rt_1 \ldots t_n)^+ & = & Rt_1 \ldots t_n \\
(\neg Rt_1 \ldots t_n)^+ & = & \neg Rt_1 \ldots t_n \\
(=(t_1, \ldots, t_n))^+ & = & \exists x_m \backslash t_1, \ldots, t_{n-1}(t_n = x_m), \text{ where} \\
 & & x_m \text{ is not free in } \varphi \\
(\neg =(t_1, \ldots, t_n))^+ & = & \exists x_0(\neg x_0 = x_0) \\
(\varphi \otimes \psi)^+ & = & \varphi^+ \vee \psi^+ \\
(\varphi \wedge \psi)^+ & = & \varphi^+ \wedge \psi^+ \\
(\exists x_n \varphi)^+ & = & \exists x_n \backslash x_{m_1}, \ldots, x_{m_k} \varphi^+, \text{ where } x_{m_1}, \ldots, x_{m_k} \\
 & & \text{are the free variables of } \varphi \text{ other than } x_n \\
(!\, x_n \varphi)^+ & = & \forall x_n \varphi^+
\end{array}$$

It is again an immediate consequence of the definitions that for all $\mathcal{M}$, all φ, and all X we have

X is of type φ if and only if $\mathcal{M} \models_X \varphi^+$ in dependence friendly logic.

The two translations $\varphi \mapsto \varphi^*$ and $\varphi \mapsto \varphi^+$ demonstrate clearly that team logic is built on top of dependence friendly logic, and a fortiori, on top of independence friendly logic. The addition team logic brings to independence friendly logic is classical negation. As this paper shows this addition calls for rather substantial revision of independence friendly logic.

It is well-known[5] that independence friendly logic can be presented in the existential fragment Σ^1_1 of second order logic. If the same presentation is applied to team logic, we go up from Σ^1_1 to the unrestricted second order logic. The proof of the following theorem is an easy adaption of the corresponding result for independence friendly logic in [Ho$_1$97a]:

Theorem 6.1. We can associate with every type $\varphi(x_{i_1}, \ldots, x_{i_n})$ in vocabulary L a second order sentence $\eta_\varphi(U)$, where U is n-ary, such that for all L-structures $\mathcal{M}$ and teams X with $\mathrm{dom}(X) = \{i_1, \ldots, i_n\}$ the following conditions are equivalent

1. X is of type φ in $\mathcal{M}$.
2. $(\mathcal{M}, X) \models \eta_\varphi(U)$.

Corollary 6.2. For every type φ there is a second order sentence η_φ such that for all models $\mathcal{M}$ we have $\mathcal{M} \models \varphi$ if and only if $\mathcal{M} \models \eta_\varphi$.

[5] For details, we refer to [Ho$_1$97a].

With the translation $\varphi \mapsto \eta_\varphi$ we can consider team logic a fragment of second order logic, even if the origin of team logic in the dependence relation $=(t_1, \ldots, t_n)$ is totally different from the origin of second order logic, and even if the logical operations of team logic are totally different from those of second order logic.

We could draw many immediate conclusions from Corollary 6.2 and from what is known about second order logic.

We have given a translation of team logic into second order logic. Now we give an implicit translation of second and higher order logic in team logic. The translation is implicit in the sense that it uses new predicates and an extension of the universe. However, the new predicates and the new universe are unique up to isomorphism. We omit the quite standard proof of the following result, which is essentially contained already in [En70] and [$\mathrm{Kr}_2\mathrm{La}_0$79].

Theorem 6.3. Suppose L is a vocabulary and $n \in \mathbb{N}$. There is a type φ in the vocabulary $L' = L \cup \{P, E\}$ such that for all L-structures $\mathcal{M}$ there is a unique (mod $\cong$) $\mathcal{N}$ of type φ with $(\mathcal{N} \restriction L)^{(P^{\mathcal{N}})} = \mathcal{M}$. Moreover, we can associate with every sentence ψ of second order logic in vocabulary L, with no second order variables of arity $> n$, a dependence type ξ_ψ in the vocabulary L' such that the following conditions are equivalent for all L-structures $\mathcal{M}$:

1. $\mathcal{M} \models \psi$
2. $\mathcal{N} \models \xi_\psi$ for the unique (mod $\cong$) $\mathcal{N}$ such that $\mathcal{N} \models \varphi$ and $(\mathcal{N} \restriction L)^{(P^{\mathcal{N}})} = \mathcal{M}$.

Corollary 6.4. A second order sentence ψ has a model if and only if $\varphi \wedge \xi_\psi$ is the type of $\{\emptyset\}$ in some model.

With the translation $\varphi \mapsto \xi_\varphi$ we can consider second order logic an implicitly defined fragment of team logic. An explicit translation, following Harel [$\mathrm{Ha}_1$79], has been constructed by Ville Nurmi.

The *decision problem* of a logic is the problem, with a sentence φ as input, whether $\mathcal{M} \models \varphi$ for all $\mathcal{M}$. The *consistency problem* of a logic is the problem, with a sentence φ as input, whether $\mathcal{M} \models \varphi$ for some $\mathcal{M}$. The *Löwenheim number* of a logic is the smallest cardinal κ such that for any sentence φ, if $\mathcal{M} \models \varphi$ for some $\mathcal{M}$, then $\mathcal{M} \models \varphi$ for some $\mathcal{M}$ of cardinality $\leq \kappa$. The *Hanf number* of a logic is the smallest cardinal κ such that for any sentence φ, if $\mathcal{M} \models \varphi$ for some $\mathcal{M}$ of cardinality $\geq \kappa$, then $\mathcal{M} \models \varphi$ for models $\mathcal{M}$ of arbitrarily large cardinality. A logic satisfies the Suslin-Kleene Interpolation Theorem if every model class which is a relativized reduct of a definable model class and whose complement has the same property is itself a definable model class. The Δ-extension of a logic is the smallest extension of it to a logic with the Suslin-Kleene Interpolation Theorem. For more details concerning these concepts we refer to [Ba$_6$Fe$_0$85].

Corollary 6.5. The decision problems of team logic and second order logic are recursively isomorphic (and Π_2-complete, see [Vä01] and Footnote 6). They have the same Löwenheim and Hanf numbers. They have the same Δ-extension.

Note that the corresponding corollary for independence friendly logic says: The decision problems of independence friendly logic and second order logic are recursively isomorphic. The consistency problem of independence friendly logic is co-r.e. while that of second order logic is Σ_2-complete[6]. The Löwenheim and Hanf number of independence friendly logic is $\aleph_0$, while the Löwenheim number of second order logic is between the first measurable and the first supercompact cardinals, if such exist, and the Hanf number of second order logic is between the first supercompact and the first extendible cardinal, if such exist (see [Ma$_0$71]).

7 Ehrenfeucht-Fraïssé game

We now introduce an Ehrenfeucht-Fraïssé game adequate for team logic and use this game to characterize team logic. This game is nothing else than a "two-directional" version of the Ehrenfeucht-Fraïssé game of independence friendly logic presented in [Vä02].

Definition 7.1. Let $\mathcal{M}$ and $\mathcal{N}$ be two structures of the same vocabulary. The game $\mathrm{EF}_n^{\mathrm{TL}}$ has two players and n moves. The position after move m is a pair (X, Y), where $X \subseteq M^{i_m}$ and $Y \subseteq N^{i_m}$ for some i_m. In the beginning the position is $(\varnothing, \varnothing)$ and $i_0 = 0$. Suppose the position after move number m is (X, Y). There are the following possibilities for the continuation of the game:

Splitting move: Player I represents X (or Y) as a union $X = X_0 \cup X_1$. Then player II represents Y (respectively, X) as a union $Y = Y_0 \cup Y_1$. Now player I chooses whether the game continues from the position (X_0, Y_0) or from the position (X_1, Y_1).

[6] We use the notation of Lévy [Lé65]. The Σ_0-formulas, which are at the same time called Π_0-formulas, are all formulas in the vocabulary $\{\in\}$ obtained from atomic formulas by the operations $\neg, \vee, \wedge$ and the *bounded quantifiers* $\exists x_0(x_0 \in x_1 \wedge \varphi)$ and $\forall x_0(x_0 \in x_1 \to \varphi)$. The Σ_{n+1}-formulas are obtained from Π_n-formulas by existential quantification. The Π_{n+1}-formulas are obtained from Σ_n-formulas by existential quantification. A set $P \subseteq \mathbb{N}$ is called Σ_n-definable if there is a Σ_n-formula $\varphi(x_0)$ of set theory such that $n \in P \iff \varphi(n)$. A problem is called Σ_n-complete if the set itself is Σ_n-definable and, moreover, for every Σ_n-definable set $X \subseteq \mathbb{N}$ there is a recursive function $f : \mathbb{N} \to \mathbb{N}$ such that $n \in X \iff f(n) \in P$. The concepts of a Π_n-definable set and a Π_n-complete set are defined analogously.

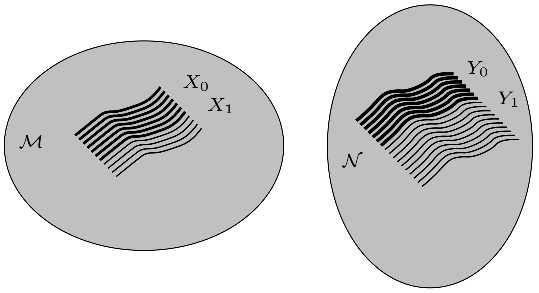


FIGURE 3. A splitting move

Duplication move: Player I decides that the game should continue from the new position

$$(X(M/i_m), Y(N/i_m)).$$

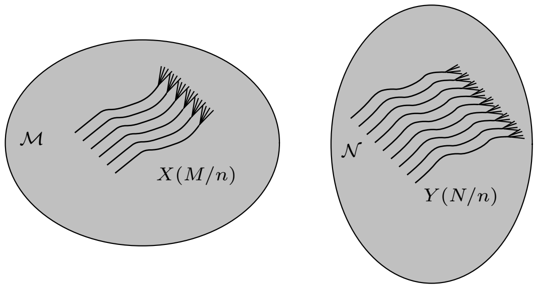


FIGURE 4. A duplication move

Supplementing move: Player I chooses a function $F : X \to M$ (or $F : Y \to N$). Then player II chooses a function $G : Y \to N$ (respectively, $G : X \to M$). Then the game continues from the position $(X(F/i_m), Y(G/i_m))$.

After n moves the position (X_n, Y_n) is reached and the game ends. Player II is the winner, if

$$X_n \text{ is of type } \varphi \text{ in } \mathcal{M} \Leftrightarrow Y_n \text{ is of type } \varphi \text{ in } \mathcal{N}$$

holds for all atomic types $\varphi(x_0, \ldots, x_{i_n-1})$. Otherwise player I wins.

This is a game of perfect information and the concept of winning strategy is defined as usual. The game is determined by the Gale-Stewart theorem.

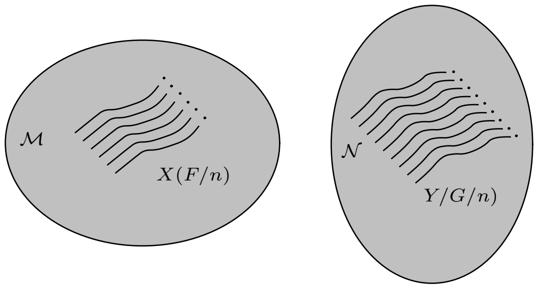


FIGURE 5. A supplementing move

Define

$$\begin{array}{lcl}
\mathrm{qr}^{\mathrm{TL}}(\varphi) & = & 0 \text{ if } \varphi \text{ is atomic,} \\
\mathrm{qr}^{\mathrm{TL}}(\varphi \otimes \psi) & = & \max(\mathrm{qr}^{\mathrm{TL}}(\varphi), \mathrm{qr}^{\mathrm{TL}}(\psi))+1, \\
\mathrm{qr}^{\mathrm{TL}}(\varphi \wedge \psi) & = & \max(\mathrm{qr}^{\mathrm{TL}}(\varphi), \mathrm{qr}^{\mathrm{TL}}(\psi)), \\
\mathrm{qr}^{\mathrm{TL}}(\exists x_n \varphi) & = & \mathrm{qr}^{\mathrm{TL}}(\varphi)+1, \\
\mathrm{qr}^{\mathrm{TL}}(!x_n \varphi) & = & \mathrm{qr}^{\mathrm{TL}}(\varphi)+1, \\
\mathrm{qr}^{\mathrm{TL}}(\sim \varphi) & = & \mathrm{qr}^{\mathrm{TL}}(\varphi).
\end{array}$$

Let Type_n^m be the set of types φ with $\mathrm{qr}^{\mathrm{TL}}(\varphi) \leq m$ and with free variables among $x_0, \ldots, x_{n-1}$. We write $\mathcal{M} \equiv_{\mathrm{TL}}^n \mathcal{N}$, if $\mathcal{M} \models \varphi$ is equivalent to $\mathcal{N} \models \varphi$ for all φ in Type_0^n, and $\mathcal{M} \equiv_{\mathrm{TL}} \mathcal{N}$ if $\mathcal{M} \equiv_{\mathrm{TL}}^n \mathcal{N}$ for all n. Note that there are for each n and m, up to logical equivalence, only finitely many types in Type_n^m.

Theorem 7.2. Suppose $\mathcal{M}$ and $\mathcal{N}$ are models of the same vocabulary. Then the following conditions are equivalent:

(1) Player II has a winning strategy in the game $\mathrm{EF}_n^{\mathrm{TL}}(\mathcal{M}, \mathcal{N})$.

(2) $\mathcal{M} \equiv_{\mathrm{TL}}^n \mathcal{N}$.

Proof. It is easy to prove by induction on m the equivalence, for all n, of the following two statements:

$(3)_m$ Player II has a winning strategy in the game $\mathrm{EF}_m^{\mathrm{TL}}(\mathcal{M}, \mathcal{N})$ in position (X, Y), where $X \subseteq M^n$ and $Y \subseteq N^n$.

$(4)_m$ If φ is a type in Type_n^m, then X is of type φ in $\mathcal{M}$ if and only if Y is of type φ in $\mathcal{N}$.

Q.E.D.

Corollary 7.3. Suppose $\mathcal{M}$ and $\mathcal{N}$ are models of the same vocabulary. Then the following conditions are equivalent:

(1) $\mathcal{M} \equiv_{\mathrm{TL}} \mathcal{N}$.

(2) For all natural numbers n, player II has a winning strategy in the game $\mathrm{EF}_n^{\mathrm{TL}}(\mathcal{M},\mathcal{N})$.

Using the fact that there are for each n and m, up to logical equivalence, only finitely many types in Type_n^m, it is easy to prove that a model class K is the class of models of a type in Type_0^n if and only if K is closed under the relation $\equiv_{\mathrm{DF}}^n$.

From this and the above theorem we get:

Corollary 7.4. Suppose K is a model class. Then the following conditions are equivalent:

(1) K is the class of models of a type of team logic.

(2) There is a natural number n such that K is closed under the relation

$$\mathcal{M}R\mathcal{N} \iff \text{Player } II \text{ has a winning strategy in } \mathrm{EF}_n^{\mathrm{TL}}(\mathcal{M},\mathcal{N}).$$

We have obtained, after all, a purely game-theoretic definition of team logic.

References

[Ar74] W.W. Armstrong. Dependency structures of data base relationships. *Information Processing* 74:580–583, 1974.

[$\mathrm{Ba}_6\mathrm{Fe}_0$85] J. Barwise & S. Feferman, eds. *Model-theoretic logics.* Springer, 1985.

[En70] H.B. Enderton. Finite partially-ordered quantifiers. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 16:393–397, 1970.

[$\mathrm{Ha}_1$79] D. Harel. Characterizing second-order logic with first-order quantifiers. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 25(5):419–422, 1979.

[$\mathrm{Hi}_1$96] J. Hintikka. *The Principles of Mathematics Revisited.* Cambridge University Press, 1996.

[$\mathrm{Ho}_1$97a] W. Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL* 5(4):539–563, 1997.

[$\mathrm{Ho}_1$97b] W. Hodges. Some strange quantifiers. In [$\mathrm{MyRo}_4\mathrm{Sa}_1$97, pp. 51–65].

[$\mathrm{Kr}_2\mathrm{La}_0$79] M. Krynicki & A.H. Lachlan. On the semantics of the Henkin quantifier. *Journal of Symbolic Logic* 44(2):184–200, 1979.

[Lé65] A. Lévy. A hierarchy of formulas in set theory. *Memoirs of the American Mathematical Society* 57:76, 1965.

[$\mathrm{Ma}_0$71] M. Magidor. On the role of supercompact and extendible cardinals in logic. *Israel Journal of Mathematics* 10:147–157, 1971.

[$\mathrm{MyRo}_4\mathrm{Sa}_1$97] J. Mycielski, G. Rozenberg & A. Salomaa, eds. *Structures in Logic and Computer Science, A Selection of Essays in Honor of Andrzej Ehrenfeucht*, *Lecture Notes in Computer Science* 1261. Springer, 1997.

[Vä01] J. Väänänen. Second-order logic and foundations of mathematics. *The Bulletin of Symbolic Logic* 7(4):504–520, 2001.

[Vä02] J. Väänänen. On the semantics of informational independence. *Logic Journal of the IGPL* 10(3):339–352, 2002.

[Vä07] J. Väänänen. *Dependence Logic: A New Approach to Independence Friendly Logic*, *London Mathematical Society Student Texts* 70. Cambridge University Press, 2007.

[$\mathrm{VäHo}_1\infty$] J. Väänänen & W. Hodges. Dependence of variables construed as an atomic formula. Preprint.

DEMO — A Demo of Epistemic Modelling[*]

Jan van Eijck

Centrum voor Wiskunde en Informatica
Kruislaan 413
1098 SJ Amsterdam, The Netherlands

Utrecht Instituut voor Linguïstiek / Onderzoeksinstituut voor Taal en Spraak
Universiteit Utrecht
P.O. Box 85253
3508 AG Utrecht, The Netherlands

Netherlands Institute for Advanced Study in the Humanities and Social Sciences
Meijboomlaan 1
2242 PR Wassenaar, The Netherlands

jve@cwi.nl

Abstract

This paper introduces and documents *DEMO*, a Dynamic Epistemic Modelling tool. *DEMO* allows modelling epistemic updates, graphical display of update results, graphical display of action models, formula evaluation in epistemic models, translation of dynamic epistemic formulas to PDL formulas. Also, *DEMO* implements the reduction of dynamic epistemic logic to PDL. The paper is an exemplar of tool building for epistemic update logic. It contains the essential code of an implementation of *DEMO* in Haskell, in Knuth's 'literate programming' style.

1 Introduction

In this introduction we shall demonstrate how *DEMO*, which is short for *Dynamic Epistemic MOdelling*,[1] can be used to check semantic intuitions about what goes on in epistemic update situations.[2] For didactic purposes,

[*] The author is grateful to the Netherlands Institute for Advanced Studies (NIAS) for providing the opportunity to complete this paper as Fellow-in-Residence. This report and the tool that it describes were prompted by a series of questions voiced by Johan van Benthem in his talk at the annual meeting of the Dutch Association for Theoretical Computer Science, in Utrecht, on March 5, 2004. Thanks to Johan van Benthem, Hans van Ditmarsch, Barteld Kooi and Ji Ruan for valuable feedback and inspiring discussion. Two anonymous referees made suggestions for improvement, which are herewith gracefully acknowledged.

[1] Or short for *DEMO of Epistemic MOdelling*, for those who prefer co-recursive acronyms.

[2] The program source code is available from http://www.cwi.nl/~jve/demo/.

the initial examples have been kept extremely simple. Although the situation of message passing about just two basic propositions with just three epistemic agents already reveals many subtleties, the reader should bear in mind that *DEMO* is capable of modelling much more complex situations.

In a situation where you and I know nothing about a particular aspect of the state of the world (about whether p and q hold, say), our state of knowledge is modelled by a Kripke model where the worlds are the four different possibilities for the truth of p and q ($\varnothing$, p, q, pq), your epistemic accessibility relation $\sim_a$ is the total relation on these four possibilities, and mine $\sim_b$ is the total relation on these four possibilities as well. There is also c, who like the two of us, is completely ignorant about p and q. This initial model is generated by *DEMO* as follows.

```
DEMO> showM (initE [P 0,Q 0] [a,b,c])
==> [0,1,2,3]
[0,1,2,3]
(0,[])(1,[p])(2,[q])(3,[p,q])
(a,[[0,1,2,3]])
(b,[[0,1,2,3]])
(c,[[0,1,2,3]])
```

Here `initE` generates an initial epistemic model, and `showM` shows that model in an appropriate form, in this case in the partition format that is made possible by the fact that the epistemic relations are all equivalences.

As an example of a different kind of representation, let us look at the picture that can be generated with *dot* [Ga$_0$Ko$_5$No$_0$06] from the file produced by the *DEMO* command `writeP "filename" (initE [P 0,Q 0])`, as represented in Figure 1.

This is a model where none of the three agents a, b or c can distinguish between the four possibilities about p and q. *DEMO* shows the partitions generated by the accessibility relations $\sim_a, \sim_b, \sim_c$. Since these three relations are total, the three partitions each consist of a single block. Call this model `e0`.

Now suppose a wants to know whether p is the case. She asks whether p and receives a truthful answer from somebody who is in a position to know. This answer is conveyed to a in a message. b and c have heard a's question, and so are aware of the fact that an answer may have reached a. b and c have seen *that* an answer was delivered, but they don't know which answer. This is not a secret communication, for b and c know that a has inquired about p. The situation now changes as follows:

```
DEMO> showM (upd e0 (message a p))
==> [1,4]
[0,1,2,3,4,5]
(0,[])(1,[p])(2,[p])(3,[q])(4,[p,q])
(5,[p,q])
```

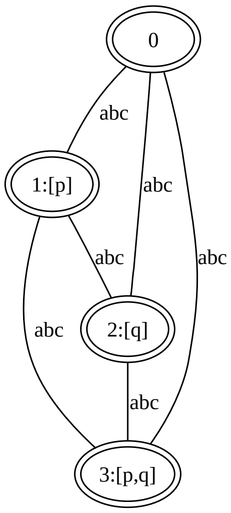


FIGURE 1.

```
(a,[[0,2,3,5],[1,4]])
(b,[[0,1,2,3,4,5]])
(c,[[0,1,2,3,4,5]])
```

Note that `upd` is a function for updating an epistemic model with (a representation of) a communicative action. In this case, the result is again a model where the three accessibility relations are equivalences, but one in which a has restricted her range of possibilities to $1, 4$ (these are worlds where p is the case), while for b and c all possibilities are still open. Note that this epistemic model has two 'actual worlds': this means that there are two possibilities that are compatible with 'how things really are'. In graphical display format these 'actual worlds' show up as double ovals, as seen in Figure 2.

DEMO also allows us to display the action models corresponding to the epistemic updates. For the present example (we have to indicate that we want the action model for the case where $\{a, b, c\}$ is the set of relevant agents):

```
showM ((message a p) [a,b,c])
==> [0]
[0,1]
(0,p)(1,
T)
(a,[[0],[1]])
```

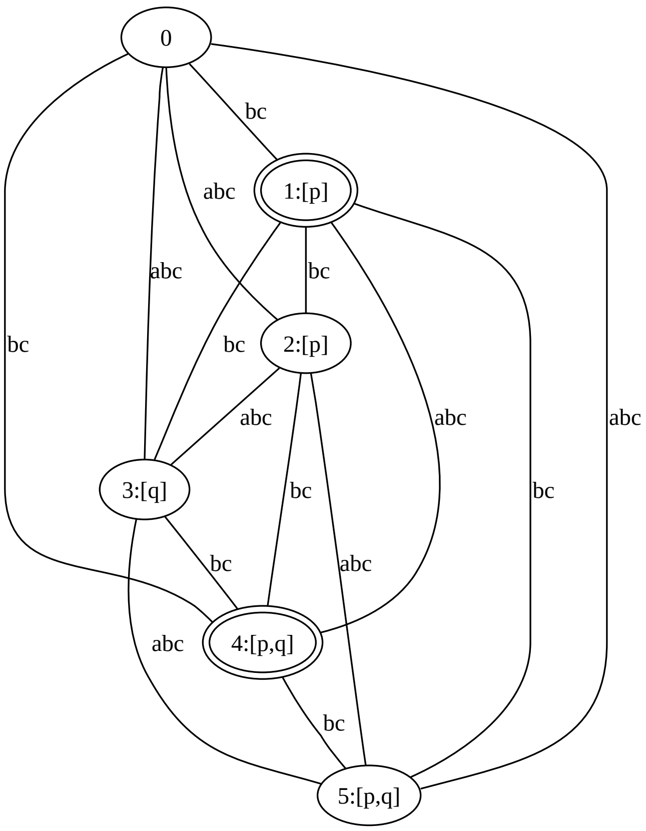


Figure 2.

```
(b,[[0,1]])
(c,[[0,1]])
```

Notice that in the result of updating the initial situation with this message, some subtle things have changed for b and c as well. Before the arrival of the message, $\Box_b(\neg\Box_a p \wedge \neg\Box_a\neg p)$ was true, for b knew that a did not know about p. But now b has heard a's question about p, and is aware of the fact that an answer has reached a. So in the new situation b knows that a knows about p. In other words, $\Box_b(\Box_a p \vee \Box_a\neg p)$ has become true. On the other hand it is still the case that b knows that a knows nothing about q: $\Box_b\neg\Box_a q$ is still true in the new situation. The situation for c is similar to that for b. These things can be checked in *DEMO* as follows:

```
DEMO> isTrue (upd e0 (message a p)) (K b (Neg (K a q)))
True
DEMO> isTrue (upd e0 (message a p)) (K b (Neg (K a p)))
False
```

If you receive the same message about p twice, the second time the message gets delivered has no further effect. Note the use of `upds` for a sequence of updates.

```
DEMO> showM (upds e0 [message a p, message a p])
==> [1,4]
[0,1,2,3,4,5]
(0,[])(1,[p])(2,[p])(3,[q])(4,[p,q])
(5,[p,q])
(a,[[0,2,3,5],[1,4]])
(b,[[0,1,2,3,4,5]])
(c,[[0,1,2,3,4,5]])
```

Now suppose that the second action is a message informing b about p:

```
DEMO> showM (upds e0 [message a p, message b p])
==> [1,6]
[0,1,2,3,4,5,6,7,8,9]
(0,[])(1,[p])(2,[p])(3,[p])(4,[p])
(5,[q])(6,[p,q])(7,[p,q])(8,[p,q])(9,[p,q])

(a,[[0,3,4,5,8,9],[1,2,6,7]])
(b,[[0,2,4,5,7,9],[1,3,6,8]])
(c,[[0,1,2,3,4,5,6,7,8,9]])
```

The graphical representation of this model is slightly more difficult to fathom at a glance. See Figure 3. In this model a and b both know about p, but they do not know about each other's knowledge about p. c still knows nothing, and both a and b know that c knows nothing. Both $\Box_a\Box_b p$ and $\Box_b\Box_a p$ are false in this model. $\Box_a\neg\Box_b p$ and $\Box_b\neg\Box_a p$ are false as well, but $\Box_a\neg\Box_c p$ and $\Box_b\neg\Box_c p$ are true.

```
DEMO> isTrue (upds e0 [message a p, message b p]) (K a (K b p))
False
DEMO> isTrue (upds e0 [message a p, message b p]) (K b (K a p))
False
DEMO> isTrue (upds e0 [message a p, message b p]) (K b (Neg (K b p)))
False
DEMO> isTrue (upds e0 [message a p, message b p]) (K b (Neg (K c p)))
True
```

The order in which a and b are informed does not matter:

```
DEMO> showM (upds e0 [message b p, message a p])
==> [1,6]
[0,1,2,3,4,5,6,7,8,9]
```

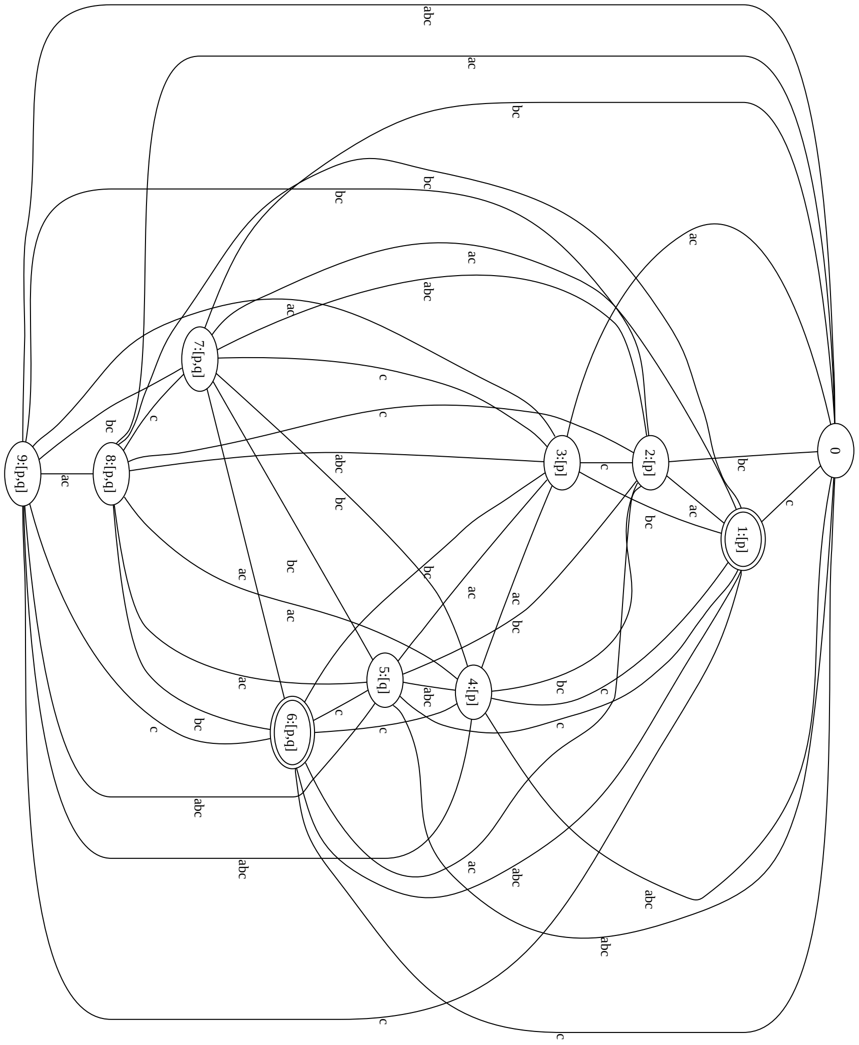


FIGURE 3. Situation after second message

```
(0,[])(1,[p])(2,[p])(3,[p])(4,[p])
(5,[q])(6,[p,q])(7,[p,q])(8,[p,q])(9,[p,q])

(a,[[0,2,4,5,7,9],[1,3,6,8]])
(b,[[0,3,4,5,8,9],[1,2,6,7]])
(c,[[0,1,2,3,4,5,6,7,8,9]])
```

Modulo renaming this is the same as the earlier result. The example shows that the epistemic effects of distributed message passing are quite different from those of a public announcement or a group message.

```
DEMO> showM (upd e0 (public p))
==> [0,1]
[0,1]
(0,[p])(1,[p,q])
(a,[[0,1]])
(b,[[0,1]])
(c,[[0,1]])
```

The result of the public announcement that p is that a, b and c are informed that p and about each other's knowledge about p.

DEMO allows to compare the action models for public announcement and individual message passing:

```
DEMO> showM ((public p) [a,b,c])
==> [0]
[0]
(0,p)
(a,[[0]])
(b,[[0]])
(c,[[0]])

DEMO> showM ((cmp [message a p, message b p, message c p]) [a,b,c])
==> [0]
[0,1,2,3,4,5,6,7]
(0,p)(1,p)(2,p)(3,p)(4,p)
(5,p)(6,p)(7,T)
(a,[[0,1,2,3],[4,5,6,7]])
(b,[[0,1,4,5],[2,3,6,7]])
(c,[[0,2,4,6],[1,3,5,7]])
```

Here `cmp` gives the sequential composition of a list of communicative actions. This involves, among other things, computation of the appropriate preconditions for the combined action model.

More subtly, the situation is also different from a situation where a, b receive the same message that p, with a being aware of the fact that b receives the message and vice versa. Such group messages create common knowledge.

```
DEMO> showM (groupM [a,b] p [a,b,c])
==> [0]
[0,1]
(0,p)(1,T)
(a,[[0],[1]])
(b,[[0],[1]])
(c,[[0,1]])
```

The difference with the case of the two separate messages is that now a and b are aware of each other's knowledge that p:

```
DEMO> isTrue (upd e0 (groupM [a,b] p)) (K a (K b p))
True
DEMO> isTrue (upd e0 (groupM [a,b] p)) (K b (K a p))
True
```

In fact, this awareness goes on, for arbitrary nestings of $\Box_a$ and $\Box_b$, which is what common knowledge means. Common knowledge can be checked directly, as follows:

```
DEMO> isTrue (upd e0 (groupM [a,b] p)) (CK [a,b] p)
True
```

It is also easily checked in *DEMO* that in the case of the separate messages no common knowledge is achieved.

Next, look at the case where two separate messages reach a and b, one informing a that p and the other informing b that $\neg q$:

```
DEMO> showM (upds e0 [message a p, message b (Neg q)])
==> [2]
[0,1,2,3,4,5,6,7,8]
(0,[])(1,[])(2,[p])(3,[p])(4,[p])
(5,[p])(6,[q])(7,[p,q])(8,[p,q])
(a,[[0,1,4,5,6,8],[2,3,7]])
(b,[[0,2,4],[1,3,5,6,7,8]])
(c,[[0,1,2,3,4,5,6,7,8]])
```

Again the order in which these messages are delivered is immaterial for the end result, as you should expect:

```
DEMO> showM (upds e0 [message b (Neg q), message a p])
==> [2]
[0,1,2,3,4,5,6,7,8]
(0,[])(1,[])(2,[p])(3,[p])(4,[p])
(5,[p])(6,[q])(7,[p,q])(8,[p,q])
(a,[[0,1,3,5,6,8],[2,4,7]])
(b,[[0,2,3],[1,4,5,6,7,8]])
(c,[[0,1,2,3,4,5,6,7,8]])
```

Modulo a renaming of worlds, this is the same as the previous result.

The logic of public announcements and private messages is related to the logic of knowledge, with [$\mathrm{Hi}_1$62] as the pioneer publication. This logic satisfies the following postulates:

- knowledge distribution $\Box_a(\varphi \Rightarrow \psi) \Rightarrow (\Box_a\varphi \Rightarrow \Box_a\psi)$ (if a knows that φ implies ψ, and she knows φ, then she also knows ψ),
- positive introspection $\Box_a\varphi \Rightarrow \Box_a\Box_a\varphi$ (if a knows φ, then a knows that she knows φ),
- negative introspection $\neg\Box_a\varphi \Rightarrow \Box_a\neg\Box_a\varphi$ (if a does not know φ, then she knows that she does not know),
- truthfulness $\Box_a\varphi \Rightarrow \varphi$ (if a knows φ then φ is true).

As is well known, the first of these is valid on all Kripke frames, the second is valid on precisely the transitive Kripke frames, the third is valid on precisely the euclidean Kripke frames (a relation R is euclidean if it satisfies $\forall x \forall y \forall z((xRy \wedge xRz) \Rightarrow yRz)$), and the fourth is valid on precisely the reflexive Kripke frames. A frame satisfies transitivity, euclideanness and reflexivity iff it is an equivalence relation, hence the logic of knowledge is the logic of the so-called S5 Kripke frames: the Kripke frames with an equivalence $\sim_a$ as epistemic accessibility relation. Multi-agent epistemic logic extends this to multi-S5, with an equivalence $\sim_b$ for every $b \in B$, where b is the set of epistemic agents.

Now suppose that instead of open messages, we use *secret* messages. If a secret message is passed to a, b and c are not even aware that any communication is going on. This is the result when a receives a secret message that p in the initial situation:

```
DEMO> showM (upd e0 (secret [a] p))
==> [1,4]
[0,1,2,3,4,5]
(0,[])(1,[p])(2,[p])(3,[q])(4,[p,q])
(5,[p,q])
(a,[([],[0,2,3,5]),([],[1,4])])
(b,[([1,4],[0,2,3,5])])
(c,[([1,4],[0,2,3,5])])
```

This is not an S5 model anymore. The accessibility for a is still an equivalence, but the accessibility for b is lacking the property of reflexivity. The worlds $1, 4$ that make up a's conceptual space (for these are the worlds accessible for a from the actual worlds $1, 4$) are precisely the worlds where the b and c arrows are *not* reflexive. b enters his conceptual space from the vantage points 1 and 4, but b does not see these vantage points itself. Similarly for c. In the *DEMO* representation, the list `([1,4],[0,2,3,5])` gives the entry points `[1,4]` into conceptual space `[0,2,3,5]`.

The secret message has no effect on what b and c believe about the facts of the world, but it has effected b's and c's beliefs about the beliefs of a in a disastrous way. These beliefs have become inaccurate. For instance, b

now believes that a does *not* know that p, but he is mistaken! The formula $\Box_b \neg \Box_a p$ is true in the actual worlds, but $\neg \Box_a p$ is false in the actual worlds, for a *does* know that p, because of the secret message. Here is what *DEMO* says about the situation (`isTrue` evaluates a formula in all of the actual worlds of an epistemic model):

```
DEMO> isTrue (upd e0 (secret [a] p)) (K b (Neg (K a p)))
True
DEMO> isTrue (upd e0 (secret [a] p)) (Neg (K a p))
False
```

This example illustrates a regress from the world of knowledge to the world of consistent belief: the result of the update with a secret propositional message does not satisfy the postulate of truthfulness anymore.

The logic of consistent belief satisfies the following postulates:

- knowledge distribution $\Box_a(\varphi \Rightarrow \psi) \Rightarrow (\Box_a \varphi \Rightarrow \Box_a \psi)$,
- positive introspection $\Box_a \varphi \Rightarrow \Box_a \Box_a \varphi$,
- negative introspection $\neg \Box_a \varphi \Rightarrow \Box_a \neg \Box_a \varphi$,
- consistency $\Box_a \varphi \Rightarrow \Diamond_a \varphi$ (if a believes that φ then there is a world where φ is true, i.e., φ is consistent).

Consistent belief is like knowledge, except for the fact that it replaces the postulate of truthfulness $\Box_a \varphi \Rightarrow \varphi$ by the weaker postulate of consistency.

Since the postulate of consistency determines the serial Kripke frames (a relation R is serial if $\forall x \exists y\, xRy$), the principles of consistent belief determine the Kripke frames that are transitive, euclidean and serial, the so-called KD45 frames.

In the conceptual world of secrecy, inconsistent beliefs are not far away. Suppose that a, after having received a secret message informing her about p, sends a message to b to the effect that $\Box_a p$. The trouble is that this is *inconsistent* with what b believes.

```
DEMO> showM (upds e0 [secret [a] p, message b (K a p)])
==> [1,5]
[0,1,2,3,4,5,6,7]
(0,[])(1,[p])(2,[p])(3,[p])(4,[q])
(5,[p,q])(6,[p,q])(7,[p,q])
(a,([],[([],[0,3,4,7]),([],[1,2,5,6])]))
(b,([1,5],[([2,6],[0,3,4,7])]))
(c,([],[([1,2,5,6],[0,3,4,7])]))
```

This is not a KD45 model anymore, for it lacks the property of seriality for b's belief relation. b's belief contains two isolated worlds $1, 5$. Since 1 is the actual world, this means that b's belief state has become inconsistent: from now on, b will believe *anything.*

So we have arrived at a still weaker logic. The logic of possibly inconsistent belief satisfies the following postulates:

- knowledge distribution $\Box_a(\varphi \Rightarrow \psi) \Rightarrow (\Box_a\varphi \Rightarrow \Box_a\psi)$,
- positive introspection $\Box_a\varphi \Rightarrow \Box_a\Box_a\varphi$,
- negative introspection $\neg\Box_a\varphi \Rightarrow \Box_a\neg\Box_a\varphi$.

This is the logic of K45 frames: frames that are transitive and euclidean.

In [vE$_1$04a] some results and a list of questions are given about the possible deterioration of knowledge and belief caused by different kind of message passing. E.g., the result of updating an S5 model with a public announcement or a non-secret message, if defined, is again S5. The result of updating an S5 model with a secret message to some of the agents, if defined, need not even be KD45. One can prove that the result is KD45 iff the model we start out with satisfies certain epistemic conditions. The update result always is K45. Such observations illustrate why S5, KD45 and K45 are ubiquitous in epistemic modelling. See [BldRVe$_1$01, Go$_0$02] for general background on modal logic, and [Ch$_3$80, Fa+95] for specific background on these systems.

If this introduction has convinced the reader that the logic of public announcements, private messages and secret communications is rich and subtle enough to justify the building of the conceptual modelling tools to be presented in the rest of the report, then it has served its purpose.

In the rest of the report, we first fix a formal version of epistemic update logic as an implementation goal. After that, we are ready for the implementation.

Further information on various aspects of dynamic epistemic logic is provided in [Ba$_4$02, Ba$_4$Mo$_3$So$_1$99, vB01b, vB06, vD00, Fa+95, Ge$_2$99a, Ko$_4$03].

2 Design

DEMO is written in a high level functional programming language Haskell [Jo$_2$03]. Haskell is a non-strict, purely-functional programming language named after Haskell B. Curry. The design is modular. Operations on lists and characters are taken from the standard Haskell `List` and `Char` modules. The following modules are part of *DEMO*:

Models The module that defines general models over a number of agents. In the present implementation these are A through E. It turns out that more than five agents are seldom needed in epistemic modelling.

General models have variables for their states and their state adornments. By letting the state adornments be valuations we get *Kripke models*, by letting them be formulas we get *update models.*

MinBis The module for minimizing models under bisimulation by means of partition refinement.

Display The module for displaying models in various formats. Not discussed in this paper.

ActEpist The module that specializes general models to action models and epistemic models. Formulas may contain action models as operators. Action models contain formulas. The definition of formulas is therefore also part of this module.

DPLL Implementation of Davis, Putnam, Logemann, Loveland (DPLL) theorem proving [$\mathrm{Da_1Lo_0Lo_4}$62, $\mathrm{Da_1Pu}$60] for propositional logic. The implementation uses discrimination trees or *tries*, following [$\mathrm{Zh_0St_5}$00]. This is used for formula simplification. Not discussed in this paper.

Semantics Implementation of the key semantic notions of epistemic update logic. It handles the mapping from communicative actions to action models.

DEMO Main module.

3 Main module

```
module DEMO
    (
     module List,
     module Char,
     module Models,
     module Display,
     module MinBis,
     module ActEpist,
     module DPLL,
     module Semantics
    )
    where

import List import Char import Models import Display import MinBis
import ActEpist import DPLL import Semantics
```

The first version of *DEMO* was written in March 2004. This version was extended in May 2004 with an implementation of automata and a translation function from epistemic update logic to Automata PDL. In September 2004, I discovered a direct reduction of epistemic update logic to PDL [$\mathrm{vE_1}$04b]. This motivated a switch to a PDL-like language, with extra modalities for action update and automata update. I decided to leave in the automata for the time being, for nostalgic reasons.

In Summer 2005, several example modules with *DEMO* programs for epistemic puzzles (some of them contributed by Ji Ruan) and for checking of security protocols (with contributions by Simona Orzan) were added, and the program was rewritten in a modular fashion.

In Spring 2006, automata update was removed, and in Autumn 2006 the code was refactored for the present report:

```
version :: String
version = "DEMO 1.06, Autumn 2006"
```

4 Definitions

4.1 Models and updates

In this section we formalize the version of dynamic epistemic logic that we are going to implement.

Let p range over a set of basic propositions P and let a range over a set of agents Ag. Then the language of PDL over P, Ag is given by:

$$\begin{array}{rcl} \varphi & ::= & \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi \\ \pi & ::= & a \mid ?\varphi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \end{array}$$

Employ the usual abbreviations: $\bot$ is shorthand for $\neg\top$, $\varphi_1 \vee \varphi_2$ is shorthand for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \to \varphi_2$ is shorthand for $\neg(\varphi_1 \wedge \varphi_2)$, $\varphi_1 \leftrightarrow \varphi_2$ is shorthand for $(\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$, and $\langle\pi\rangle\varphi$ is shorthand for $\neg[\pi]\neg\varphi$. Also, if $B \subseteq Ag$ and B is finite, use B as shorthand for $b_1 \cup b_2 \cup \cdots$. Under this convention, formulas for expressing general knowledge $E_B\varphi$ take the shape $[B]\varphi$, while formulas for expressing common knowledge $C_B\varphi$ appear as $[B^*]\varphi$, i.e., $[B]\varphi$ expresses that it is general knowledge among agents B that φ, and $[B^*]\varphi$ expresses that it is common knowledge among agents B that φ. In the special case where $B = \varnothing$, B turns out equivalent to $?\bot$, the program that always fails.

The semantics of PDL over P, Ag is given relative to labelled transition systems $\mathbf{M} = (W, V, R)$, where W is a set of worlds (or states), $V : W \to \mathcal{P}(P)$ is a valuation function, and $R = \{\xrightarrow{a} \subseteq W \times W \mid a \in Ag\}$ is a set of labelled transitions, i.e., binary relations on W, one for each label a. In what follows, we shall take the labelled transitions for a to represent the epistemic alternatives of an agent a.

The formulae of PDL are interpreted as subsets of $W_\mathbf{M}$ (the state set of $\mathbf{M}$), the actions of PDL as binary relations on $W_\mathbf{M}$, as follows:

$$\begin{array}{rcl} [\![\top]\!]^\mathbf{M} & = & W_\mathbf{M} \\ [\![p]\!]^\mathbf{M} & = & \{w \in W_\mathbf{M} \mid p \in V_\mathbf{M}(w)\} \end{array}$$

$$\begin{array}{rcl}
[\![\neg\varphi]\!]^{\mathbf{M}} & = & W_{\mathbf{M}} - [\![\varphi]\!]^{\mathbf{M}} \\
[\![\varphi_1 \wedge \varphi_2]\!]^{\mathbf{M}} & = & [\![\varphi_1]\!]^{\mathbf{M}} \cap [\![\varphi_2]\!]^{\mathbf{M}} \\
[\![[\pi]\varphi]\!]^{\mathbf{M}} & = & \{w \in W_{\mathbf{M}} \mid \forall v(\text{ if } (w,v) \in [\![\pi]\!]^{\mathbf{M}} \text{ then } v \in [\![\varphi]\!]^{\mathbf{M}})\}
\end{array}$$

$$\begin{array}{rcl}
[\![a]\!]^{\mathbf{M}} & = & \xrightarrow{a}_{\mathbf{M}} \\
[\![?\varphi]\!]^{\mathbf{M}} & = & \{(w,w) \in W_{\mathbf{M}} \times W_{\mathbf{M}} \mid w \in [\![\varphi]\!]^{\mathbf{M}}\} \\
[\![\pi_1;\pi_2]\!]^{\mathbf{M}} & = & [\![\pi_1]\!]^{\mathbf{M}} \circ [\![\pi_2]\!]^{\mathbf{M}} \\
[\![\pi_1 \cup \pi_2]\!]^{\mathbf{M}} & = & [\![\pi_1]\!]^{\mathbf{M}} \cup [\![\pi_2]\!]^{\mathbf{M}} \\
[\![\pi^*]\!]^{\mathbf{M}} & = & ([\![\pi]\!]^{\mathbf{M}})^*
\end{array}$$

If $w \in W_{\mathbf{M}}$ then we use $\mathbf{M} \models_w \varphi$ for $w \in [\![\varphi]\!]^{\mathbf{M}}$. The paper [Ba$_4Mo_3So_1$03] proposes to model epistemic actions as epistemic models, with valuations replaced by preconditions. See also: [vB01b, vB06, vD00, vE$_1$04b, Fa+95, Ge$_2$99a, Ko$_4$03, Ru$_0$04].

Action models for a given language $\mathcal{L}$. Let a set of agents Ag and an epistemic language $\mathcal{L}$ be given. An action model for $\mathcal{L}$ is a triple $A = ([s_0,\ldots,s_{n-1}], \text{pre}, T)$ where $[s_0,\ldots,s_{n-1}]$ is a finite list of action states, $\text{pre} : \{s_0,\ldots,s_{n-1}\} \to \mathcal{L}$ assigns a precondition to each action state, and $T : Ag \to \mathcal{P}(\{s_0,\ldots,s_{n-1}\}^2)$ assigns an accessibility relation $\xrightarrow{a}$ to each agent $a \in Ag$.

A pair $\mathbf{A} = (A, s)$ with $s \in \{s_0,\ldots,s_{n-1}\}$ is a pointed action model, where s is the action that actually takes place.

The list ordering of the action states in an action model will play an important role in the definition of the program transformations associated with the action models.

In the definition of action models, $\mathcal{L}$ can be any language that can be interpreted in PDL models. Actions can be executed in PDL models by means of the following product construction:

Action Update. Let a PDL model $\mathbf{M} = (W, V, R)$, a world $w \in W$, and a pointed action model (A, s), with $A = ([s_0,\ldots,s_{n-1}], \text{pre}, T)$, be given. Suppose $w \in [\![\text{pre}(s)]\!]^{\mathbf{M}}$. Then the result of executing (A, s) in $(\mathbf{M}, w)$ is the model $(\mathbf{M} \otimes A, (w, s))$, with $\mathbf{M} \otimes A = (W', V', R')$, where

$$\begin{array}{rcl}
W' & = & \{(w,s) \mid s \in \{s_0,\ldots,s_{n-1}\}, w \in [\![\text{pre}(s)]\!]^{\mathbf{M}}\} \\
V'(w,s) & = & V(w) \\
R'(a) & = & \{((w,s),(w',s')) \mid (w,w') \in R(a), (s,s') \in T(a)\}.
\end{array}$$

In case there is a set of actual worlds and a set of actual actions, the definition is similar: those world/action pairs survive where the world satisfies the preconditions of the action (see below).

The language of $\mathrm{PDL}^{\mathrm{DEL}}$ (update PDL) is given by extending the PDL language with update constructions $[A, s]\varphi$, where (A, s) is a pointed action model. The interpretation of $[A, s]\varphi$ in $\mathbf{M}$ is given by:

$$[\![[A, s]\varphi]\!]^{\mathbf{M}} = \{w \in W_{\mathbf{M}} \mid \text{if } \mathbf{M} \models_w \mathrm{pre}(s) \text{ then } (w, s) \in [\![\varphi]\!]^{\mathbf{M}\otimes A}\}.$$

Using $\langle A, s\rangle\varphi$ as shorthand for $\neg[A, s]\neg\varphi$, we see that the interpretation for $\langle A, s\rangle\varphi$ turns out as:

$$[\![\langle A, s\rangle\varphi]\!]^{\mathbf{M}} = \{w \in W_{\mathbf{M}} \mid \mathbf{M} \models_w \mathrm{pre}(s) \text{ and } (w, s) \in [\![\varphi]\!]^{\mathbf{M}\otimes A}\}.$$

Updating with multiple pointed update actions is also possible. A multiple pointed action is a pair (A, S), with A an action model, and S a subset of the state set of A. Extend the language with updates $[A, S]\varphi$, and interpret this as follows:

$$[\![[A, S]\varphi]\!]^{\mathbf{M}} = \{w \in W_{\mathbf{M}} \mid \forall s \in S(\text{if } \mathbf{M} \models_w \mathrm{pre}(s) \\ \text{then } \mathbf{M} \otimes A \models_{(w,s)} \varphi)\}.$$

In [vE$_1$04b] it is shown how dynamic epistemic logic can be reduced to PDL by program transformation. Each action model $\mathbf{A}$ has associated program transformers $T^{\mathbf{A}}_{ij}$ for all states s_i, s_j in the action model, such that the following hold:

Lemma 4.1 (Program Transformation, Van Eijck [vE$_1$04b])**.** Assume A has n states $s_0, \ldots, s_{n-1}$. Then:

$$\mathbf{M} \models_w [A, s_i][\pi]\varphi \text{ iff } \mathbf{M} \models_w \bigwedge_{j=0}^{n-1} [T^{A}_{ij}(\pi)][A, s_j]\varphi.$$

This lemma allows a reduction of dynamic epistemic logic to PDL, a reduction that we shall implement in the code below.

4.2 Operations on action models

Sequential Composition. If $(\mathbf{A}, S)$ and $(\mathbf{B}, T)$ are multiple pointed action models, their sequential composition $(\mathbf{A}, S) \odot (\mathbf{B}, T)$ is given by:

$$(\mathbf{A}, S) \odot (\mathbf{B}, T) := ((W, \mathrm{pre}, R), S \times T),$$

where

- $W = W_{\mathbf{A}} \times W_{\mathbf{B}}$,
- $\mathrm{pre}(s, t) = \mathrm{pre}(s) \wedge \langle \mathbf{A}, S\rangle \mathrm{pre}(t)$,
- R is given by: $(s, t) \xrightarrow{a} (s', t') \in R$ iff $s \xrightarrow{a} s' \in R_{\mathbf{A}}$ and $t \xrightarrow{a} t' \in R_{\mathbf{B}}$.

The unit element for this operation is the action model

$$\mathbf{1} = ((\{0\}, 0 \mapsto \top, \{0 \xrightarrow{a} 0 \mid a \in Ag\}), \{0\}).$$

Updating an arbitrary epistemic model $\mathbf{M}$ with $\mathbf{1}$ changes nothing.

Non-deterministic Sum. The non-deterministic sum $\oplus$ of multiple pointed action models $(\mathbf{A}, S)$ and $(\mathbf{B}, T)$ is the action model $(\mathbf{A}, S) \oplus (\mathbf{B}, T)$ is given by:

$$(\mathbf{A}, S) \oplus (\mathbf{B}, T) := ((W, \text{pre}, R), S \uplus T),$$

where $\uplus$ denotes disjoint union, and where

- $W = W_{\mathbf{A}} \uplus W_{\mathbf{B}}$,
- $\text{pre} = \text{pre}_{\mathbf{A}} \uplus \text{pre}_{\mathbf{B}}$,
- $R = R_{\mathbf{A}} \uplus R_{\mathbf{B}}$.

The unit element for this operation is called **0**: the multiple pointed action model given by $((\varnothing, \varnothing, \varnothing), \varnothing)$.

4.3 Logics for communication

Here are some specific action models that can be used to define various languages of communication.

In order to model a **public announcement of** φ, we use the action model $(\mathbf{S}, \{0\})$ with

$$S_{\mathbf{S}} = \{0\}, p_{\mathbf{S}} = 0 \mapsto \varphi, R_{\mathbf{S}} = \{0 \xrightarrow{a} 0 \mid a \in A\}.$$

If we wish to model an **individual message to** b **that** φ, we consider the action model $(\mathbf{S}, \{0\})$ with $S_{\mathbf{S}} = \{0, 1\}$, $p_{\mathbf{S}} = 0 \mapsto \varphi, 1 \mapsto \top$, and $R_{\mathbf{S}} = \{0 \xrightarrow{b} 0, 1 \xrightarrow{b} 1\} \cup \{0 \sim_a 1 \mid a \in A - \{b\}\}$; similarly, for a **group message to** B **that** φ, we use the action model $(\mathbf{S}, \{0\})$ with

$$S_{\mathbf{S}} = \{0, 1\}, p_{\mathbf{S}} = 0 \mapsto \varphi, 1 \mapsto \top, R_{\mathbf{S}} = \{0 \sim_a 1 \mid a \in A - B\}.$$

A **secret individual communication to** b **that** φ is modelled by $(\mathbf{S}, \{0\})$ with

$$\begin{aligned} S_{\mathbf{S}} &= \{0, 1\}, \\ p_{\mathbf{S}} &= 0 \mapsto \varphi, 1 \mapsto \top, \\ R_{\mathbf{S}} &= \{0 \xrightarrow{b} 0\} \cup \{0 \xrightarrow{a} 1 \mid a \in A - \{b\}\} \cup \{1 \xrightarrow{a} 1 \mid a \in A\}, \end{aligned}$$

and a **secret group communication to** B **that** φ by $(\mathbf{S}, \{0\})$ with

$$\begin{aligned} S_{\mathbf{S}} &= \{0, 1\}, \\ p_{\mathbf{S}} &= 0 \mapsto \varphi, 1 \mapsto \top, \\ R_{\mathbf{S}} &= \{0 \xrightarrow{b} 0 \mid b \in B\} \cup \{0 \xrightarrow{a} 1 \mid a \in A - B\} \cup \{1 \xrightarrow{a} 1 \mid a \in A\}. \end{aligned}$$

We model a **test of** φ by the action model $(\mathbf{S}, \{0\})$ with

$$S_{\mathbf{S}} = \{0,1\}, p_{\mathbf{S}} = 0 \mapsto \varphi, 1 \mapsto \top, R_{\mathbf{S}} = \{0 \xrightarrow{a} 1 \mid a \in A\} \cup \{1 \xrightarrow{a} 1 \mid a \in A\},$$

an **individual revelation to** b **of a choice from** $\{\varphi_1, \ldots, \varphi_n\}$ by the action model $(\mathbf{S}, \{1, \ldots, n\})$ with

$$\begin{array}{rcl} S_{\mathbf{S}} & = & \{1, \ldots, n\}, \\ p_{\mathbf{S}} & = & 1 \mapsto \varphi_1, \ldots, n \mapsto \varphi_n, \\ R_{\mathbf{S}} & = & \{s \xrightarrow{b} s \mid s \in S_{\mathbf{S}}\} \cup \{s \xrightarrow{a} s' \mid s, s' \in S_{\mathbf{S}}, a \in A - \{b\}\}, \end{array}$$

and a **group revelation to** B **of a choice from** $\{\varphi_1, \ldots, \varphi_n\}$ by the action model $(\mathbf{S}, \{1, \ldots, n\})$ with

$$\begin{array}{rcl} S_{\mathbf{S}} & = & \{1, \ldots, n\}, \\ p_{\mathbf{S}} & = & 1 \mapsto \varphi_1, \ldots, n \mapsto \varphi_n, \\ R_{\mathbf{S}} & = & \{s \xrightarrow{b} s \mid s \in S_{\mathbf{S}}, b \in B\} \cup \{s \xrightarrow{a} s' \mid s, s' \in S_{\mathbf{S}}, a \in A - B\}. \end{array}$$

Finally, **transparent informedness of** B **about** φ is represented by the action model $(\mathbf{S}, \{0,1\})$ with $S_{\mathbf{S}} = \{0,1\}$, $p_{\mathbf{S}} = 0 \mapsto \varphi, 1 \mapsto \neg\varphi$, $R_{\mathbf{S}} = \{0 \xrightarrow{a} 0 \mid a \in A\} \cup \{0 \xrightarrow{a} 1 \mid a \in A - B\} \cup \{1 \xrightarrow{a} 0 \mid a \in A - B\} \cup \{1 \xrightarrow{a} 1 \mid a \in A\}$. Transparent informedness of B about φ is the special case of a group revelation of B of a choice from $\{\varphi, \neg\varphi\}$. Note that all but the revelation action models and the transparent informedness action models are single pointed (their sets of actual states are singletons).

On the syntactic side, we now define the corresponding languages. The language for the logic of group announcements is defined by:

$$\begin{array}{rcl} \varphi & ::= & \top \mid p \mid \neg\varphi \mid \bigwedge[\varphi_1, \ldots, \varphi_n] \mid \bigvee[\varphi_1, \ldots, \varphi_n] \mid \Box_a \varphi \\ & & \mid E_B \varphi \mid C_B \varphi \mid \; [\pi]\varphi \\ \\ \pi & ::= & \mathbf{1} \mid \mathbf{0} \mid \text{public } B\ \varphi \mid \odot[\pi_1, \ldots, \pi_n] \mid \oplus[\pi_1, \ldots, \pi_n] \end{array}$$

We use the semantics of **1**, **0**, **public** $B\ \varphi$, and the operations on multiple pointed action models from Section 4.2. For the logic of tests and group announcements, we allow tests $?\varphi$ as basic programs and add the appropriate semantics. For the logic of individual messages, the basic actions are messages to individual agents. In order to give it a semantics, we start out from the semantics of **message** $a\ \varphi$. Finally, the logic of tests, group announcements, and group revelations is as above, but now also allowing revelations from alternatives. For the semantics, we use the semantics of **reveal** $B\ \{\varphi_1, \ldots, \varphi_n\}$.

5 Kripke models

```
module Models where

import List
```

5.1 Agents

```
data Agent = A | B | C | D | E deriving (Eq,Ord,Enum,Bounded)
```

Give the agents appropriate names:

```
a, alice, b, bob, c, carol, d, dave, e, ernie :: Agent
a = A; alice = A
b = B; bob   = B
c = C; carol = C
d = D; dave  = D
e = E; ernie = E
```

Make agents showable in an appropriate way:

```
instance Show Agent where
  show A = "a"; show B = "b"; show C = "c"; show D = "d" ; show E = "e"
```

5.2 Model datatype

It will prove useful to generalize over states. We first define general models, and then specialize to action models and epistemic models. In the following definition, `state` and `formula` are variables over types. We assume that each model carries a list of distinguished states.

```
data Model state formula = Mo
                           [state]
                           [(state,formula)]
                           [Agent]
                           [(Agent,state,state)]
                           [state]
                           deriving (Eq,Ord,Show)
```

Decomposing a pointed model into a list of single-pointed models:

```
decompose ::  Model state formula -> [Model state formula]
decompose (Mo states pre agents rel points) =
  [ Mo states pre agents rel [point] | point <- points ]
```

It is useful to be able to map the precondition table to a function. Here is a general tool for that. Note that the resulting function is partial; if the function argument does not occur in the table, the value is undefined.

```
table2fct :: Eq a => [(a,b)] -> a -> b
table2fct t = \ x -> maybe undefined id (lookup x t)
```

Another useful utility is a function that creates a partition out of an equivalence relation:

```
rel2part :: (Eq a) => [a] -> (a -> a -> Bool) -> [[a]]
rel2part [] r = []
rel2part (x:xs) r = xblock : rel2part rest r
  where
  (xblock,rest) = partition (\ y -> r x y) (x:xs)
```

The *domain* of a model is its list of states:

```
domain :: Model state formula -> [state]
domain (Mo states _ _ _ _) = states
```

The *eval* of a model is its list of state/formula pairs:

```
eval :: Model state formula -> [(state,formula)]
eval (Mo _ pre _ _ _) = pre
```

The *agentList* of a model is its list of agents:

```
agentList :: Model state formula -> [Agent]
agentList (Mo _ _ ags _ _) = ags
```

The *access* of a model is its labelled transition component:

```
access :: Model state formula -> [(Agent,state,state)]
access (Mo _ _ _ rel _) = rel
```

The distinguished points of a model:

```
points :: Model state formula -> [state]
points (Mo _ _ _ _ pnts) = pnts
```

When we are looking at models, we are only interested in generated submodels, with as their domain the distinguished state(s) plus everything that is reachable by an accessibility path.

```
gsm :: Ord state => Model state formula -> Model state formula
gsm (Mo states pre ags rel points) = (Mo states' pre' ags rel' points)
   where
   states' = closure rel ags points
   pre'    = [(s,f)     | (s,f)     <- pre,
                          elem s states'                 ]
   rel'    = [(ag,s,s') | (ag,s,s') <- rel,
                          elem s states',
                          elem s' states'                ]
```

The closure of a state list, given a relation and a list of agents:

```
closure ::  Ord state =>
             [(Agent,state,state)] -> [Agent] -> [state] -> [state]
closure rel agents xs
  | xs' == xs = xs
  | otherwise = closure rel agents xs'
     where
     xs' = (nub . sort) (xs ++ (expand rel agents xs))
```

The expansion of a relation R given a state set S and a set of agents B is given by $\{t \mid s \xrightarrow{b} t \in R, s \in S, b \in B\}$. This is implemented as follows:

```
expand :: Ord state =>
            [(Agent,state,state)] -> [Agent] -> [state] -> [state]
expand rel agnts ys =
      (nub . sort . concat)
         [ alternatives rel ag state | ag    <- agnts,
                                       state <- ys       ]
```

The epistemic alternatives for agent a in state s are the states in sR_a (the states reachable through R_a from s):

```
alternatives :: Eq state =>
                 [(Agent,state,state)] -> Agent -> state -> [state]
alternatives rel ag current =
  [ s' | (a,s,s') <- rel, a == ag, s == current ]
```

6 Model minimization under bisimulation

```
module MinBis where

import List
import Models
```

6.1 Partition refinement

Any Kripke model can be simplified by replacing each state s by its bisimulation class $[s]$. The problem of finding the smallest Kripke model modulo bisimulation is similar to the problem of minimizing the number of states in a finite automaton [$\text{Ho}_4$71]. We will use partition refinement, in the spirit of [$\text{Pa}_1\text{Ta}_0$87]. Here is the algorithm:

- Start out with a partition of the state set where all states with the same precondition function are in the same class. The equality relation to be used to evaluate the precondition function is given as a parameter to the algorithm.

- Given a partition Π, for each block b in Π, partition b into sub-blocks such that two states s, t of b are in the same sub-block iff for all agents a it holds that s and t have $\xrightarrow{a}$ transitions to states in the same block of Π. Update Π to Π' by replacing each b in Π by the newly found set of sub-blocks for b.

- Halt as soon as $\Pi = \Pi'$.

Looking up and checking of two formulas against a given equivalence relation:

```
lookupFs :: (Eq a,Eq b) =>
      a -> a -> [(a,b)] -> (b -> b -> Bool) -> Bool
lookupFs i j table r = case lookup i table of
  Nothing -> lookup j table == Nothing
  Just f1 -> case lookup j table of
      Nothing -> False
      Just f2 -> r f1 f2
```

The following computes the initial partition, using a particular relation for equivalence of formulas:

```
initPartition :: (Eq a, Eq b) => Model a b -> (b -> b -> Bool) -> [[a]]
initPartition (Mo states pre ags rel points) r =
  rel2part states (\ x y -> lookupFs x y pre r)
```

Refining a partition:

```
refinePartition :: (Eq a, Eq b) =>
                     Model a b -> [[a]] -> [[a]]
refinePartition m p = refineP m p p
  where
  refineP :: (Eq a, Eq b) => Model a b -> [[a]] -> [[a]] -> [[a]]
  refineP m part [] = []
  refineP m part (block:blocks) =
     newblocks ++ (refineP m part blocks)
       where
         newblocks =
           rel2part block (\ x y -> sameAccBlocks m part x y)
```

The following is a function that checks whether two states have the same accessible blocks under a partition:

```
sameAccBlocks :: (Eq a, Eq b) =>
         Model a b -> [[a]] -> a -> a -> Bool
sameAccBlocks m@(Mo states pre ags rel points) part s t =
    and [ accBlocks m part s ag == accBlocks m part t  ag |
                                             ag <- ags ]
```

The accessible blocks for an agent from a given state, given a model and a partition can be determined by `accBlocks`:

```
accBlocks :: (Eq a, Eq b) =>
               Model a b -> [[a]] -> a -> Agent -> [[a]]
accBlocks m@(Mo states pre ags rel points) part s ag =
    nub [ bl part y | (ag',x,y) <- rel, ag' == ag, x == s ]
```

The block of an object in a partition:

```
bl :: Eq a => [[a]] -> a -> [a]
bl part x = head (filter (elem x) part)
```

Initializing and refining a partition:

```
initRefine :: (Eq a, Eq b) =>
                Model a b -> (b -> b -> Bool) -> [[a]]
initRefine m r = refine m (initPartition m r)
```

The refining process:

```
refine :: (Eq a, Eq b) => Model a b -> [[a]] -> [[a]]
refine m part = if rpart == part
                       then part
                       else refine m rpart
  where rpart = refinePartition m part
```

6.2 Minimization

We now use this to construct the minimal model. Notice the dependence on relational parameter r.

```
minimalModel :: (Eq a, Ord a, Eq b, Ord b) =>
                 (b -> b -> Bool) -> Model a b -> Model [a] b
minimalModel r m@(Mo states pre ags rel points) =
  (Mo states' pre' ags rel' points')
     where
     partition = initRefine m r
     states'   = partition
     f         = bl partition
     rel'      = (nub.sort) (map (\ (x,y,z) -> (x, f y, f z)) rel)
     pre'      = (nub.sort) (map (\ (x,y)   -> (f x, y))      pre)
     points'   =  map f points
```

Converting a's into integers, using their position in a given list of a's.

```
convert :: (Eq a, Show a) => [a] -> a  -> Integer
convert = convrt 0
  where
  convrt :: (Eq a, Show a) => Integer -> [a] -> a -> Integer
  convrt n []     x = error (show x ++ " not in list")
  convrt n (y:ys) x | x == y    = n
                    | otherwise = convrt (n+1) ys x
```

Converting an object of type Model a b into an object of type Model Integer b:

```
conv ::  (Eq a, Show a) =>
           Model a b -> Model Integer b
conv  (Mo worlds val ags acc points) =
      (Mo (map f worlds)
          (map (\ (x,y)   -> (f x, y)) val)
           ags
          (map (\ (x,y,z) -> (x, f y, f z)) acc))
          (map f points)
  where f = convert worlds
```

Use this to rename the blocks into integers:

```
bisim ::  (Eq a, Ord a, Show a, Eq b, Ord b) =>
           (b -> b -> Bool) -> Model a b -> Model Integer b
bisim r = conv . (minimalModel r)
```

7 Formulas, action models and epistemic models

```
module ActEpist where

import List
import Models
import MinBis
import DPLL
```

Module `List` is a standard Haskell module. Module `Models` is described in Chapter 5, and Module `MinBis` in Chapter 6. Module `DPLL` refers to an implementation of Davis, Putnam, Logemann, Loveland (DPLL) theorem proving (not included in this document, but available at `http://www.cwi.nl/~jve/demo`).

7.1 Formulas

Basic propositions:

```
data Prop = P Int | Q Int | R Int deriving (Eq,Ord)
```

Show these in the standard way, in lower case, with index 0 omitted.

```
instance Show Prop where
  show (P 0) = "p"; show (P i) = "p" ++ show i
  show (Q 0) = "q"; show (Q i) = "q" ++ show i
  show (R 0) = "r"; show (R i) = "r" ++ show i
```

Formulas, according to the definition:

$$\begin{array}{lcl}\varphi & ::= & \top \mid p \mid \neg\varphi \mid \bigwedge[\varphi_1,\dots,\varphi_n] \mid \bigvee[\varphi_1,\dots,\varphi_n] \mid [\pi]\varphi \mid \ [\mathbf{A}]\varphi \\ \pi & ::= & a \mid B \mid ?\varphi \mid \bigcirc[\pi_1,\dots,\pi_n] \mid \bigcup[\pi_1,\dots,\pi_n] \mid \pi^*\end{array}$$

Here, p ranges over basic propositions, a ranges over agents, B ranges over non-empty sets of agents, and $\mathbf{A}$ is a multiple pointed action model (see below) $\bigcirc$ denotes sequential composition of a list of programs. We will often write $\bigcirc[\pi_1,\pi_2]$ as $\pi_1;\pi_2$, and $\bigcup[\pi_1,\pi_2]$ as $\pi_1 \cup \pi_2$.

Note that general knowledge among agents B that φ is expressed in this language as $[B]\varphi$, and common knowledge among agents B that φ as $[B^*]\varphi$. Thus, $[B]\varphi$ can be viewed as shorthand for $[\bigcup_{b\in B} b]\varphi$. In case $B = \varnothing$, $[B]\varphi$ turns out to be equivalent to $[?\bot]\varphi$.

For convenience, we have also left in the more traditional way of expressing individual knowledge $\Box_a\varphi$, general knowledge $E_B\varphi$ and common knowledge $C_B\varphi$.

```
data Form = Top
          | Prop Prop
          | Neg  Form
          | Conj [Form]
```

```
                | Disj [Form]
                | Pr Program Form
                | K Agent Form
                | EK [Agent] Form
                | CK [Agent] Form
                | Up AM Form
                deriving (Eq,Ord)

data Program = Ag Agent
             | Ags [Agent]
             | Test Form
             | Conc [Program]
             | Sum  [Program]
             | Star Program
             deriving (Eq,Ord)
```

Some useful abbreviations:

```
impl :: Form -> Form -> Form
impl form1 form2 = Disj [Neg form1, form2]

equiv :: Form -> Form -> Form
equiv form1 form2 = Conj [form1 `impl` form2, form2 `impl` form1]

xor :: Form -> Form -> Form
xor x y = Disj [ Conj [x, Neg y], Conj [Neg x, y]]
```

The negation of a formula:

```
negation :: Form -> Form
negation (Neg form) = form
negation form       = Neg form
```

Show formulas in the standard way:

```
instance Show Form where
  show Top = "T" ; show (Prop p) = show p; show (Neg f) = '-':(show f);
  show (Conj fs)     = '&': show fs
  show (Disj fs)     = 'v': show fs
  show (Pr p f)      = '[': show p ++ "]" ++ show f
  show (K agent f)   = '[': show agent ++ "]" ++ show f
  show (EK agents f) = 'E': show agents ++ show f
  show (CK agents f) = 'C': show agents ++ show f
  show (Up pam f)    = 'A': show (points pam) ++ show f
```

Show programs in a standard way:

```
instance Show Program where
  show (Ag a)       = show a
  show (Ags as)     = show as
  show (Test f)     = '?': show f
  show (Conc ps)    = 'C': show ps
  show (Sum ps)     = 'U': show ps
  show (Star p)     = '(': show p ++ ")*"
```

Programs can get very unwieldy very quickly. As is well known, there is no normalisation procedure for regular expressions. Still, here are some rewriting steps for simplification of programs:

$$\begin{array}{lcrlcr}
\varnothing & \to & ?\bot & ?\varphi_1 \cup ?\varphi_2 & \to & ?(\varphi_1 \vee \varphi_2) \\
?\bot \cup \pi & \to & \pi & \pi \cup ?\bot & \to & \pi \\
\bigcup[] & \to & ?\bot & \bigcup[\pi] & \to & \pi \\
?\varphi_1 ; ?\varphi_2 & \to & ?(\varphi_1 \wedge \varphi_2) & ?\top ; \pi & \to & \pi \\
\pi ; ?\top & \to & \pi & ?\bot ; \pi & \to & ?\bot \\
\pi ; ?\bot & \to & ?\bot & \bigcirc[] & \to & ?\top \\
\bigcirc[\pi] & \to & \pi & (?\varphi)^* & \to & ?\top \\
(?\varphi \cup \pi)^* & \to & \pi^* & (\pi \cup ?\varphi)^* & \to & \pi^* \\
\pi^{**} & \to & \pi^*, & & &
\end{array}$$

and the $k + m + n$-ary rewriting steps

$$\bigcup[\pi_1, \ldots, \pi_k, \bigcup[\pi_{k+1}, \ldots, \pi_{k+m}], \pi_{k+m+1}, \ldots, \pi_{k+m+n}] \\ \to \quad \bigcup[\pi_1, \ldots, \pi_{k+m+n}]$$

and

$$\bigcirc\,[\pi_1, \ldots, \pi_k, \bigcirc[\pi_{k+1}, \ldots, \pi_{k+m}], \pi_{k+m+1}, \ldots, \pi_{k+m+n}] \\ \to \quad \bigcirc[\pi_1, \ldots, \pi_{k+m+n}].$$

Simplifying unions by splitting up in test part, accessibility part and rest:

```
splitU :: [Program] -> ([Form],[Agent],[Program])
splitU [] = ([],[],[])
splitU (Test f: ps) = (f:fs,ags,prs)
                       where (fs,ags,prs) = splitU ps
splitU (Ag x: ps) = (fs,union [x] ags,prs)
                       where (fs,ags,prs) = splitU ps
splitU (Ags xs: ps) = (fs,union xs ags,prs)
                       where (fs,ags,prs) = splitU ps
splitU (Sum ps: ps') = splitU (union ps ps')
splitU (p:ps) = (fs,ags,p:prs)
                       where (fs,ags,prs) = splitU ps
```

Simplifying compositions:

```
comprC :: [Program] -> [Program]
comprC [] = []
comprC (Test Top: ps)          = comprC ps
comprC (Test (Neg Top):ps)     = [Test (Neg Top)]
comprC (Test f: Test f': rest) = comprC (Test (canonF (Conj [f,f'])):
                                     rest)
comprC (Conc ps : ps')         = comprC (ps ++ ps')
comprC (p:ps)                  = let ps' = comprC ps
                                 in if ps' == [Test (Neg Top)]
                                    then [Test (Neg Top)] else p: ps'
```

Use this in the code for program simplification:

```
simpl :: Program -> Program
simpl (Ag x)                    = Ag x
simpl (Ags [])                  = Test (Neg Top)
simpl (Ags [x])                 = Ag x
simpl (Ags xs)                  = Ags xs
simpl (Test f)                  = Test (canonF f)
```

Simplifying unions:

```
simpl (Sum prs) =
  let (fs,xs,rest) = splitU (map simpl prs)
      f            = canonF (Disj fs)
  in
    if xs == [] && rest == [] then Test f
    else if xs == [] && f == Neg Top && length rest == 1
      then (head rest)
    else if xs == [] && f == Neg Top then Sum rest
    else if xs == []
      then Sum (Test f: rest)
    else if length xs == 1  && f == Neg Top
      then Sum (Ag (head xs): rest)
    else if length xs == 1 then Sum (Test f: Ag (head xs): rest)
    else if f == Neg Top then Sum (Ags xs: rest)
    else Sum (Test f: Ags xs: rest)
```

Simplifying sequential compositions:

```
simpl (Conc prs) =
    let prs' = comprC (map simpl prs)
    in
      if prs'== []                  then Test Top
      else if length prs' == 1      then head prs'
      else if head prs' == Test Top then Conc (tail prs')
      else                               Conc prs'
```

Simplifying stars:

```
simpl (Star pr) = case simpl pr of
    Test f              -> Test Top
    Sum [Test f, pr']   -> Star pr'
    Sum (Test f: prs')  -> Star (Sum prs')
    Star pr'            -> Star pr'
    pr'                 -> Star pr'
```

Property of being a purely propositional formula:

```
pureProp ::  Form -> Bool
pureProp Top       = True
pureProp (Prop _)  = True
pureProp (Neg f)   = pureProp f
pureProp (Conj fs) = and (map pureProp fs)
pureProp (Disj fs) = and (map pureProp fs)
pureProp  _        = False
```

Some example formulas and formula-forming operators:

```
bot, p0, p, p1, p2, p3, p4, p5, p6 :: Form
bot = Neg Top
p0 = Prop (P 0); p = p0; p1  = Prop (P 1); p2 = Prop (P 2)
p3 = Prop (P 3); p4 = Prop (P 4); p5 = Prop (P 5); p6 = Prop (P 6)

q0, q, q1, q2, q3, q4, q5, q6 :: Form
q0 = Prop (Q 0); q = q0; q1 = Prop (Q 1); q2 = Prop (Q 2);
q3 = Prop (Q 3); q4 = Prop (Q 4); q5 = Prop (Q 5); q6 = Prop (Q 6)

r0, r, r1, r2, r3, r4, r5, r6:: Form
r0 = Prop (R 0); r = r0; r1 = Prop (R 1); r2 = Prop (R 2)
r3 = Prop (R 3); r4 = Prop (R 4); r5 = Prop (R 5); r6 = Prop (R 6)

u   = Up ::  AM -> Form -> Form

nkap = Neg (K a p)
nkanp = Neg (K a (Neg p))
nka_p = Conj [nkap,nkanp]
```

7.2 Reducing formulas to canonical form

For computing bisimulations, it is useful to have some notion of equivalence (however crude) for the logical language. For this, we reduce formulas to a canonical form. We will derive canonical forms that are unique up to propositional equivalence, employing a propositional reasoning engine. This is still rather crude, for any modal formula will be treated as a propositional literal. The DPLL (Davis, Putnam, Logemann, Loveland) engine expects clauses represented as lists of integers, so we first have to translate to this format. This translation should start with computing a mapping from positive literals to integers. For the non-propositional operators we use a little bootstrapping, by putting the formula inside the operator in canonical form, using the function `canonF` to be defined below. Also, since the non-propositional operators all behave as Box modalities, we can reduce $\Box\top$ to $\top$:

```
mapping :: Form -> [(Form,Integer)]
mapping f = zip lits [1..k]
  where
  lits = (sort . nub . collect) f
  k    = toInteger (length lits)
  collect :: Form -> [Form]
  collect Top         = []
  collect (Prop p)    = [Prop p]
  collect (Neg f)     = collect f
  collect (Conj fs)   = concat (map collect fs)
  collect (Disj fs)   = concat (map collect fs)
  collect (Pr pr f)   = if canonF f == Top
                        then [] else [Pr pr (canonF f)]
  collect (K ag f)    = if canonF f == Top
```

```
                            then [] else [K ag (canonF f)]
    collect (EK ags f)  = if canonF f == Top
                            then [] else [EK ags (canonF f)]
    collect (CK ags f)  = if canonF f == Top
                            then [] else [CK ags (canonF f)]
    collect (Up pam f)  = if canonF f == Top
                            then [] else [Up pam (canonF f)]
```

The following code corresponds to putting in clausal form, given a mapping for the literals, and using bootstrapping for formulas in the scope of a non-propositional operator. Note that $\Box\top$ is reduced to $\top$, and $\neg\Box\top$ to $\bot$.

```
cf :: (Form -> Integer) -> Form ->
[[Integer]]
cf g (Top)            = []
cf g (Prop p)         = [[g (Prop p)]]
cf g (Pr pr f)        = if canonF f == Top then []
                        else [[g (Pr pr (canonF f))]]
cf g (K ag f)         = if canonF f == Top then []
                        else [[g (K ag (canonF f))]]
cf g (EK ags f)       = if canonF f == Top then []
                        else [[g (EK ags (canonF f))]]
cf g (CK ags f)       = if canonF f == Top then []
                        else [[g (CK ags (canonF f))]]
cf g (Up am f)        = if canonF f == Top then []
                        else [[g (Up am (canonF f))]]
cf g (Conj fs)        = concat (map (cf g) fs)
cf g (Disj fs)        = deMorgan (map (cf g) fs)
```

Negated formulas:

```
cf g (Neg Top)        = [[]]
cf g (Neg (Prop p))   = [[- g (Prop p)]]
cf g (Neg (Pr pr f))  = if canonF f == Top then [[]]
                        else [[- g (Pr pr (canonF f))]]
cf g (Neg (K ag f))   = if canonF f == Top then [[]]
                        else [[- g (K ag (canonF f))]]
cf g (Neg (EK ags f)) = if canonF f == Top then [[]]
                        else [[- g (EK ags (canonF f))]]
cf g (Neg (CK ags f)) = if canonF f == Top then [[]]
                        else [[- g (CK ags (canonF f))]]
cf g (Neg (Up am f))  = if canonF f == Top then [[]]
                        else [[- g (Up am (canonF f))]]
cf g (Neg (Conj fs))  = deMorgan (map (\ f -> cf g (Neg f)) fs)
cf g (Neg (Disj fs))  = concat   (map (\ f -> cf g (Neg f)) fs)
cf g (Neg (Neg f))    = cf g f
```

In order to explain the function `deMorgan`, we recall De Morgan's disjunction distribution which is the logical equivalence of the following expressions:

$$\varphi \vee (\psi_1 \wedge \cdots \wedge \psi_n) \leftrightarrow (\varphi \vee \psi_1) \wedge \cdots \wedge (\varphi \vee \psi_n).$$

Now the following is the code for De Morgan's disjunction distribution (for the case of a disjunction of a list of clause sets):

```
deMorgan :: [[[Integer]]] -> [[Integer]]
deMorgan [] = [[]]
deMorgan [cls] = cls
deMorgan (cls:clss) = deMorg cls (deMorgan clss)
  where
  deMorg :: [[Integer]] -> [[Integer]] -> [[Integer]]
  deMorg cls1 cls2 = (nub . concat) [ deM cl cls2 | cl <- cls1 ]
  deM :: [Integer] -> [[Integer]]  -> [[Integer]]
  deM cl cls = map (fuseLists cl) cls
```

Function `fuseLists` keeps the literals in the clauses ordered.

```
fuseLists :: [Integer] -> [Integer] -> [Integer]
fuseLists [] ys = ys
fuseLists xs [] = xs
fuseLists (x:xs) (y:ys) | abs x < abs y  = x:(fuseLists xs (y:ys))
                        | abs x == abs y = if x == y
                                           then x:(fuseLists xs ys)
                                           else if x > y
                                             then x:y:(fuseLists xs ys)
                                             else y:x:(fuseLists xs ys)
                        | abs x > abs y  = y:(fuseLists (x:xs) ys)
```

Given a mapping for the positive literals, the satisfying valuations of a formula can be collected from the output of the DPLL process. Here `dp` is the function imported from the module `DPLL`.

```
satVals :: [(Form,Integer)] -> Form -> [[Integer]]
satVals t f = (map fst . dp) (cf (table2fct t) f)
```

Two formulas are propositionally equivalent if they have the same sets of satisfying valuations, computed on the basis of a literal mapping for their conjunction:

```
propEquiv :: Form -> Form -> Bool
propEquiv f1 f2 = satVals g f1 == satVals g f2
  where g = mapping (Conj [f1,f2])
```

A formula is a (propositional) contradiction if it is propositionally equivalent to `Neg Top`, or equivalently, to `Disj []`:

```
contrad :: Form -> Bool
contrad f = propEquiv f (Disj [])
```

A formula is (propositionally) consistent if it is not a propositional contradiction:

```
consistent :: Form -> Bool
consistent = not . contrad
```

Use the set of satisfying valuations to derive a canonical form:

```
canonF :: Form -> Form
canonF f = if (contrad (Neg f))
             then Top
             else if fs == []
             then Neg Top
             else if length fs == 1
             then head fs
             else Disj fs
  where g   = mapping f
        nss = satVals g f
        g'  = \ i -> head [ form | (form,j) <- g, i == j ]
        h   = \ i -> if i < 0 then Neg (g' (abs i)) else g' i
        h'  = \ xs -> map h xs
        k   = \ xs -> if xs == []
                        then Top
                        else if length xs == 1
                                then head xs
                                else Conj xs
        fs  = map k (map h' nss)
```

This gives:

```
ActEpist> canonF p
p
ActEpist> canonF (Conj [p,Top])
p
ActEpist> canonF (Conj [p,q,Neg r])
&[p,q,-r]
ActEpist> canonF (Neg (Disj [p,(Neg p)]))
-T
ActEpist> canonF (Disj [p,q,Neg r])
v[p,&[-p,q],&[-p,-q,-r]]
ActEpist> canonF (K a (Disj [p,q,Neg r]))
[a]v[p,&[-p,q],&[-p,-q,-r]]
ActEpist> canonF (Conj  [p, Conj [q,Neg r]])
&[p,q,-r]
ActEpist> canonF (Conj  [p, Disj [q,Neg (K a (Disj []))]])
v[&[p,q],&[p,-q,-[a]-T]]
ActEpist> canonF (Conj  [p, Disj [q,Neg (K a (Conj []))]])
&[p,q]
```

7.3 Action models and epistemic models

Action models and epistemic models are built from states. We assume states are represented by integers:

```
type State = Integer
```

Epistemic models are models where the states are of type `State`, and the precondition function assigns lists of basic propositions (this specializes the precondition function to a valuation).

```
type EM = Model State [Prop]
```

Find the valuation of an epistemic model:

```
valuation  :: EM -> [(State,[Prop])]
valuation = eval
```

Action models are models where the states are of type `State`, and the precondition function assigns objects of type `Form`. The only difference between an action model and a static model is in the fact that action models have a precondition function that assigns a formula instead of a set of basic propositions.

```
type AM = Model State Form
```

The preconditions of an action model:

```
preconditions :: AM -> [Form]
preconditions (Mo states pre ags acc points) =
   map (table2fct pre) points
```

Sometimes we need a single precondition:

```
precondition :: AM -> Form
precondition am = canonF (Conj (preconditions am))
```

The zero action model **0**:

```
zero :: [Agent] -> AM
zero ags = (Mo [] [] ags [] [])
```

The purpose of action models is to define relations on the class of all static models. States with precondition $\perp$ can be pruned from an action model. For this we define a specialized version of the `gsm` function:

```
gsmAM :: AM -> AM
gsmAM (Mo states pre ags acc points) =
  let
    points' = [ p | p <- points, consistent (table2fct pre p) ]
    states' = [ s | s <- states, consistent (table2fct pre s) ]
    pre'    = filter (\ (x,_) -> elem x states') pre
    f       = \ (_,s,t) -> elem s states' && elem t states'
    acc'    = filter f acc
  in
  if points' == []
     then zero ags
     else gsm (Mo states' pre' ags acc' points')
```

7.4 Program transformation

For every action model A with states $s_0, \dots, s_{n-1}$ we define a set of n^2 program transformers $T^A_{i,j}$ $(0 \leq i < n, 0 \leq j < n)$, as follows [vE$_1$04b]:

$$\begin{aligned}
T^A_{ij}(a) &= \begin{cases} ?\mathrm{pre}(s_i); a & \text{if } s_i \xrightarrow{a} s_j, \\ ?\bot & \text{otherwise} \end{cases} \\
T^A_{ij}(?\varphi) &= \begin{cases} ?(\mathrm{pre}(s_i) \wedge [A, s_i]\varphi) & \text{if } i = j, \\ ?\bot & \text{otherwise} \end{cases} \\
T^A_{ij}(\pi_1; \pi_2) &= \bigcup_{k=0}^{n-1} (T^A_{ik}(\pi_1); T^A_{kj}(\pi_2)) \\
T^A_{ij}(\pi_1 \cup \pi_2) &= T^A_{ij}(\pi_1) \cup T^A_{ij}(\pi_2) \\
T^A_{ij}(\pi^*) &= K^A_{ijn}(\pi)
\end{aligned}$$

where $K^A_{ijk}(\pi)$ is a (transformed) program for all the π^* paths from s_i to s_j that can be traced through A while avoiding a pass through intermediate states s_k and higher. Thus, $K^A_{ijn}(\pi)$ is a program for all the π^* paths from s_i to s_j that can be traced through A, period.

$K^A_{ijk}(\pi)$ is defined by recursing on k, as follows:

$$K^A_{ij0}(\pi) = \begin{cases} ?\top \cup T^A_{ij}(\pi) & \text{if } i = j, \\ T^A_{ij}(\pi) & \text{otherwise} \end{cases}$$

$$K^A_{ij(k+1)}(\pi) = \begin{cases} (K^A_{kkk}(\pi))^* & \text{if } i = k = j, \\ (K^A_{kkk}(\pi))^*; K^A_{kjk}(\pi) & \text{if } i = k \neq j, \\ K^A_{ikk}(\pi); (K^A_{kkk}(\pi))^* & \text{if } i \neq k = j, \\ K^A_{ijk}(\pi) \cup (K^A_{ikk}(\pi); (K^A_{kkk}(\pi))^*; K^A_{kjk}(\pi)) & \text{otherwise.} \end{cases}$$

Lemma 7.1 (Kleene Path)**.** Suppose $(w, w') \in [\![T^A_{ij}(\pi)]\!]^{\mathbf{M}}$ iff there is a π path from (w, s_i) to (w', s_j) in $\mathbf{M} \otimes A$. Then $(w, w') \in [\![K^A_{ijn}(\pi)]\!]^{\mathbf{M}}$ iff there is a π^* path from (w, s_i) to (w', s_j) in $\mathbf{M} \otimes A$.

The Kleene path lemma is the key ingredient in the proof of the following program transformation lemma.

Lemma 7.2 (Program Transformation)**.** Assume A has n states $s_0, \dots, s_{n-1}$. Then:

$$\mathbf{M} \models_w [A, s_i][\pi]\varphi \text{ iff } \mathbf{M} \models_w \bigwedge_{j=0}^{n-1} [T^A_{ij}(\pi)][A, s_j]\varphi.$$

The implementation of the program transformation functions is given here:

```
transf :: AM -> Integer -> Integer -> Program -> Program
transf am@(Mo states pre allAgs acc points) i j (Ag ag) =
   let
     f = table2fct pre i
   in
   if elem (ag,i,j) acc && f == Top          then Ag ag
   else if elem (ag,i,j) acc && f /= Neg Top then Conc [Test f, Ag ag]
   else Test (Neg Top)
transf am@(Mo states pre allAgs acc points) i j (Ags ags) =
   let ags' = nub [ a | (a,k,m) <- acc, elem a ags, k == i, m == j ]
       ags1 = intersect ags ags'
       f    = table2fct pre i
   in
     if ags1 == [] || f == Neg Top        then Test (Neg Top)
     else if f == Top && length ags1 == 1 then Ag (head ags1)
     else if f == Top                     then Ags ags1
     else Conc [Test f, Ags ags1]
transf am@(Mo states pre allAgs acc points) i j (Test f) =
   let
     g = table2fct pre i
   in
   if i == j
      then Test (Conj [g,(Up am f)])
      else Test (Neg Top)
transf am@(Mo states pre allAgs acc points) i j (Conc [])  =
  transf am i j (Test Top)
transf am@(Mo states pre allAgs acc points) i j (Conc [p]) =
  transf am i j p
transf am@(Mo states pre allAgs acc points) i j (Conc (p:ps)) =
  Sum [ Conc [transf am i k p, transf am k j (Conc ps)] | k <- [0..n] ]
    where n = toInteger (length states - 1)
transf am@(Mo states pre allAgs acc points) i j (Sum [])  =
  transf am i j (Test (Neg Top))
transf am@(Mo states pre allAgs acc points) i j (Sum [p]) =
  transf am i j p
transf am@(Mo states pre allAgs acc points) i j (Sum ps) =
  Sum [ transf am i j p | p <- ps ]
transf am@(Mo states pre allAgs acc points) i j (Star p) =
  kleene am i j n p
    where n = toInteger (length states)
```

The following is the implementation of $K_{ijk}^{\mathbf{A}}$:

```
kleene ::  AM -> Integer -> Integer -> Integer -> Program -> Program
kleene am i j 0 pr =
  if i == j
    then Sum [Test Top, transf am i j pr]
    else transf am i j pr
kleene am i j k pr
  | i == j && j == pred k = Star (kleene am i i i pr)
  | i == pred k           =
    Conc [Star (kleene am i i i pr), kleene am i j i pr]
    | j == pred k          =
      Conc [kleene am i j j pr, Star (kleene am j j j pr)]
    | otherwise            =
        Sum [kleene am i j k' pr,
             Conc [kleene am i k' k' pr,
                   Star (kleene am k' k' k' pr), kleene am k' j k' pr]]
        where k' = pred k
```

Transformation plus simplification:

```
tfm ::  AM -> Integer -> Integer -> Program -> Program
tfm am i j pr = simpl (transf am i j pr)
```

The program transformations can be used to translate Update PDL to PDL, as follows:

$$\begin{array}{rclrcl} t(\top) & = & \top & t(p) & = & p \\ t(\neg\varphi) & = & \neg t(\varphi) & t(\varphi_1 \wedge \varphi_2) & = & t(\varphi_1) \wedge t(\varphi_2) \\ t([\pi]\varphi) & = & [r(\pi)]t(\varphi) & t([A,s]\top) & = & \top \end{array}$$

$$\begin{array}{rcl} t([A,s]p) & = & t(\text{pre}(s)) \to p \\ t([A,s]\neg\varphi) & = & t(\text{pre}(s)) \to \neg t([A,s]\varphi) \\ t([A,s](\varphi_1 \wedge \varphi_2)) & = & t([A,s]\varphi_1) \wedge t([A,s]\varphi_2) \\ t([A,s_i][\pi]\varphi) & = & \bigwedge_{j=0}^{n-1}[T_{ij}^A(r(\pi))]t([A,s_j]\varphi) \\ t([A,s][A',s']\varphi) & = & t([A,s]t([A',s']\varphi)) \\ t([A,S]\varphi) & = & \bigwedge_{s\in S} t[A,s]\varphi) \end{array}$$

$$\begin{array}{rclrcl} r(a) & = & a & r(B) & = & B \\ r(?\varphi) & = & ?t(\varphi) & r(\pi_1;\pi_2) & = & r(\pi_1);r(\pi_2) \\ r(\pi_1 \cup \pi_2) & = & r(\pi_1) \cup r(\pi_2) & r(\pi^*) & = & (r(\pi))^*. \end{array}$$

The correctness of this translation follows from direct semantic inspection, using the program transformation lemma for the translation of formulas of type $[A,s_i][\pi]\varphi$.

The crucial clauses in this translation procedure are those for formulas of the forms $[A,S]\varphi$ and $[A,s]\varphi$, and more in particular the one for formulas of the form $[A,s][\pi]\varphi$. It makes sense to give separate functions for the steps that pull the update model through program π given formula φ.

```
step0, step1 :: AM -> Program -> Form -> Form
step0 am@(Mo states pre allAgs acc []) pr f = Top
step0 am@(Mo states pre allAgs acc [i]) pr f = step1 am pr f
step0 am@(Mo states pre allAgs acc  is) pr f =
  Conj [ step1 (Mo states pre allAgs acc [i]) pr f | i <- is ]
step1 am@(Mo states pre allAgs acc [i]) pr f =
   Conj [ Pr (transf am i j (rpr pr))
             (Up (Mo states pre allAgs acc [j]) f) | j <- states ]
```

Perform a single step, and put in canonical form:

```
step :: AM -> Program -> Form -> Form
step am pr f = canonF (step0 am pr f)

t :: Form -> Form
t Top = Top
t (Prop p) = Prop p
t (Neg f) = Neg (t f)
t (Conj fs) = Conj (map t fs)
t (Disj fs) = Disj (map t fs)
t (Pr pr f) = Pr (rpr pr) (t f)
t (K x f) = Pr (Ag x) (t f)
t (EK xs f)  = Pr (Ags xs) (t f)
t (CK xs f)  = Pr (Star (Ags xs)) (t f)
```

Translations of formulas starting with an action model update:

```
t (Up am@(Mo states pre allAgs acc [i]) f) = t' am f
t (Up am@(Mo states pre allAgs acc  is) f)  =
   Conj [ t' (Mo states pre allAgs acc [i]) f | i <- is ]
```

Translations of formulas starting with a single pointed action model update are performed by `t'`:

```
t' :: AM -> Form -> Form
t' am Top            = Top
t' am (Prop p)       = impl (precondition am) (Prop p)
t' am (Neg f)        =  Neg (t' am f)
t' am (Conj fs)      = Conj (map (t' am) fs)
t' am (Disj fs)      = Disj (map (t' am) fs)
t' am (K x f)        = t' am (Pr (Ag x) f)
t' am (EK xs f)      = t' am (Pr (Ags xs) f)
t' am (CK xs f)      = t' am (Pr (Star (Ags xs)) f)
t' am (Up am'f)      = t' am (t (Up am' f))
```

The crucial case is an update action having scope over a program. We may assume that the update action is single pointed.

```
t' am@(Mo states pre allAgs acc [i]) (Pr pr f) =
   Conj [ Pr (transf am i j (rpr pr))
              (t' (Mo states pre allAgs acc [j]) f) | j <- states ]
t' am@(Mo states pre allAgs acc is) (Pr pr f) =
   error "action model not single pointed"
```

Translations for programs:

```
rpr :: Program -> Program
rpr (Ag x)       = Ag x
rpr (Ags xs)     = Ags xs
rpr (Test f)     = Test (t f)
rpr (Conc ps)    = Conc (map rpr ps)
rpr (Sum  ps)    = Sum  (map rpr ps)
rpr (Star p)     = Star (rpr p)
```

Translating and putting in canonical form:

```
tr :: Form -> Form
tr = canonF . t
```

Some example translations:

```
ActEpist> tr (Up (public p) (Pr (Star (Ags [b,c])) p))
T
ActEpist> tr (Up (public (Disj [p,q])) (Pr (Star (Ags [b,c])) p))
[(U[?T,C[?v[p,q],[b,c]]])*]v[p,&[-p,-q]]
ActEpist> tr (Up (groupM [a,b] p) (Pr (Star (Ags [b,c])) p))
[C[C[(U[?T,C[?p,[b,c]]])*,C[?p,[c]]],(U[U[?T,[b,c]],
        C[c,(U[?T,C[?p,[b,c]]])*,C[?p,[c]]]]])*]]p
ActEpist> tr (Up (secret [a,b] p) (Pr (Star (Ags [b,c])) p))
[C[C[(U[?T,C[?p,[b]]])*,C[?p,[c]]],(U[U[?T,[b,c]],
        C[?-T,(U[?T,C[?p,[b]]])*,C[?p,[c]]]]])*]]p
```

8 Semantics

```
module Semantics
where

import List
import Char
import Models
import Display
import MinBis
import ActEpist
import DPLL
```

8.1 Semantics implementation

The group alternatives of group of agents a are the states that are reachable through $\bigcup_{a \in A} R_a$.

```
groupAlts :: [(Agent,State,State)] -> [Agent] -> State -> [State]
groupAlts rel agents current =
  (nub . sort . concat) [ alternatives rel a current | a <- agents ]
```

The common knowledge alternatives of group of agents a are the states that are reachable through a finite number of R_a links, for $a \in A$.

```
commonAlts :: [(Agent,State,State)] -> [Agent] -> State -> [State]
commonAlts rel agents current =
  closure rel agents (groupAlts rel agents current)
```

The model update function takes a static model and and action model and returns an object of type `Model (State,State) [Prop]`. The up function takes an epistemic model and an action model and returns an epistemic model. Its states are the `(State,State)` pairs that result from the cartesian product construction described in [$\text{Ba}_4\text{Mo}_3\text{So}_1$99]. Note that the update function uses the truth definition (given below as `isTrueAt`).

We will set up matters in such way that updates with action models get their list of agents from the epistemic model that gets updated. For this, we define:

```
type FAM = [Agent] -> AM

up :: EM -> FAM -> Model (State,State) [Prop]
up m@(Mo worlds val ags acc points) fam =
   Mo worlds' val' ags acc' points'
   where
   am@(Mo states pre _ susp actuals) = fam ags
   worlds' = [ (w,s) | w <- worlds, s <- states,
                       formula <- maybe [] (\ x -> [x]) (lookup s pre),
                       isTrueAt w m formula                        ]
   val'    = [ ((w,s),props) | (w,props) <- val,
                                s        <- states,
                                elem (w,s) worlds'                 ]
   acc'    = [ (ag1,(w1,s1),(w2,s2)) | (ag1,w1,w2) <- acc,
                                        (ag2,s1,s2) <- susp,
                                         ag1 == ag2,
                                         elem (w1,s1) worlds',
                                         elem (w2,s2) worlds'      ]
   points' = [ (p,a) | p <- points, a <- actuals,
                       elem (p,a) worlds'                          ]
```

An action model is tiny if its action list is empty or a singleton list:

```
tiny :: FAM -> Bool
tiny fam = length actions <= 1
  where actions = domain (fam [minBound..maxBound])
```

The appropriate notion of equivalence for the base case of the bisimulation for epistemic models is "having the same valuation".

```
sameVal :: [Prop] -> [Prop] -> Bool
sameVal ps qs = (nub . sort) ps ==  (nub . sort) qs
```

Bisimulation minimal version of generated submodel of update result for epistemic model and pointed action models:

```
upd :: EM -> FAM -> EM
upd sm fam = if tiny fam then conv (up sm fam)
             else bisim (sameVal) (up sm fam)
```

Non-deterministic update with a list of pointed action models:

```
upds  :: EM -> [FAM] -> EM
upds = foldl upd
```

At last we have all ingredients for the truth definition.

```
isTrueAt :: State -> EM -> Form -> Bool
isTrueAt w m Top = True
isTrueAt w m@(Mo worlds val ags acc pts) (Prop p) =
  elem p (concat [ props | (w',props) <- val, w'==w ])
isTrueAt w m (Neg f)   = not (isTrueAt w m f)
isTrueAt w m (Conj fs) = and (map (isTrueAt w m) fs)
isTrueAt w m (Disj fs) = or  (map (isTrueAt w m) fs)
```

The clauses for individual knowledge, general knowledge and common knowledge use the functions `alternatives`, `groupAlts` and `commonAlts` to compute the relevant accessible worlds:

```
isTrueAt w m@(Mo worlds val ags acc pts) (K ag f) =
  and (map (flip ((flip isTrueAt) m) f) (alternatives acc ag w))
isTrueAt w m@(Mo worlds val ags acc pts) (EK agents f) =
  and (map (flip ((flip isTrueAt) m) f) (groupAlts acc agents w))
isTrueAt w m@(Mo worlds val ags acc pts) (CK agents f) =
  and (map (flip ((flip isTrueAt) m) f) (commonAlts acc agents w))
```

In the clause for $[\mathbf{M}]\varphi$, the result of updating the static model M with action model $\mathbf{M}$ may be undefined, but in this case the precondition $P(s_0)$ of the designated state s_0 of $\mathbf{M}$ will fail in the designated world w_0 of M. By making the clause for $[\mathbf{M}]\varphi$ check for $M \models_{w_0} P(s_0)$, truth can be defined as a total function.

```
isTrueAt w m@(Mo worlds val ags rel pts) (Up am f) =
 and [ isTrue  m' f |
       m' <- decompose (upd (Mo worlds val ags rel [w]) (\ ags -> am))]
```

Checking for truth in *all* the designated points of an epistemic model:

```
isTrue :: EM -> Form -> Bool
isTrue (Mo worlds val ags rel pts) form =
   and [ isTrueAt w (Mo worlds val ags rel pts) form | w <- pts ]
```

8.2 Tools for constructing epistemic models

The following function constructs an initial epistemic model where the agents are completely ignorant about their situation, as described by a list of basic propositions. The input is a list of basic propositions used for constructing the valuations.

```
initE :: [Prop] -> [Agent] -> EM
initE allProps ags = (Mo worlds val ags accs points)
  where
    worlds = [0..(2^k - 1)]
    k      = length allProps
    val    = zip worlds (sortL (powerList allProps))
    accs   = [ (ag,st1,st2) | ag  <- ags,
                              st1 <- worlds,
                              st2 <- worlds      ]
    points = worlds
```

This uses the following utilities:

```
powerList  :: [a] -> [[a]]
powerList  [] = [[]]
powerList  (x:xs) = (powerList xs) ++ (map (x:) (powerList xs))

sortL :: Ord a => [[a]] -> [[a]]
sortL  = sortBy (\ xs ys -> if length xs < length ys then LT
                               else if length xs > length ys then GT
                               else compare xs ys)
```

Some initial models:

```
e00 :: EM
e00 = initE [P 0] [a,b]

e0 :: EM
e0 = initE [P 0,Q 0] [a,b,c]
```

8.3 From communicative actions to action models

Computing the update for a public announcement:

```
public :: Form -> FAM
public form ags =
   (Mo [0] [(0,form)] ags [ (a,0,0) | a <- ags ] [0])
```

Public announcements are S5 models:

```
DEMO> showM (public p [a,b,c])
==> [0]
[0]
(0,p)
(a,[[0]])
(b,[[0]])
(c,[[0]])
```

Computing the update for passing a group announcement to a list of agents: the other agents may or may not be aware of what is going on. In the limit case where the message is passed to all agents, the message is a public announcement.

```
groupM :: [Agent] -> Form -> FAM
groupM gr form agents =
  if sort gr == sort agents
    then public form agents
    else
      (Mo
         [0,1]
         [(0,form),(1,Top)]
         agents
         ([ (a,0,0) | a <- agents ]
           ++ [ (a,0,1) | a <- agents \\ gr ]
           ++ [ (a,1,0) | a <- agents \\ gr ]
           ++ [ (a,1,1) | a <- agents          ])
         [0])
```

Group announcements are S5 models:

```
Semantics> showM (groupM [a,b] p [a,b,c,d,e])
=> [0]
[0,1]
(0,p)(1,T)
(a,[[0],[1]])
(b,[[0],[1]])
(c,[[0,1]])
(d,[[0,1]])
(e,[[0,1]])
```

Computing the update for an individual message to b that φ:

```
message :: Agent -> Form -> FAM
message agent = groupM [agent]
```

Another special case of a group message is a test. Tests are updates that messages to the empty group:

```
test :: Form -> FAM
test = groupM []
```

Computing the update for passing a *secret* message to a list of agents: the other agents remain unaware of the fact that something goes on. In the limit case where the secret is divulged to all agents, the secret becomes a public update.

```
secret :: [Agent] -> Form -> FAM
secret agents form all_agents =
  if sort agents == sort all_agents
    then public form agents
    else
      (Mo
         [0,1]
         [(0,form),(1,Top)]
          all_agents
         ([ (a,0,0) | a <- agents ]
           ++ [ (a,0,1) | a <- all_agents \\ agents ]
           ++ [ (a,1,1) | a <- all_agents            ])
         [0])
```

Secret messages are KD45 models:

```
DEMO> showM (secret [a,b] p [a,b,c])
==> [0]
[0,1]
(0,p)(1,T)
(a,[([],[0]),([],[1])])
(b,[([],[0]),([],[1])])
(c,[([0],[1])])
```

Here is a multiple pointed action model for the communicative action of revealing one of a number of alternatives to a list of agents, in such a way that it is common knowledge that one of the alternatives gets revealed (in [$\mathrm{Ba_4Mo_3So_1}$03] this is called *common knowledge of alternatives*).

```
reveal :: [Agent] -> [Form] -> FAM
reveal ags forms all_agents =
  (Mo
     states
     (zip states forms)
     all_agents
     ([ (ag,s,s) | s <- states, ag <- ags ]
       ++
      [ (ag,s,s') | s <- states, s' <- states, ag <- others ])
     states)
    where states = map fst (zip [0..] forms)
          others = all_agents \\ ags
```

Here is an action model for the communication that reveals to a one of p_1, q_1, r_1.

```
Semantics> showM (reveal [a] [p1,q1,r1] [a,b])
==> [0,1,2]
[0,1,2]
(0,p1)(1,q1)(2,r1)
(a,[[0],[1],[2]])
(b,[[0,1,2]])
```

A group of agents B gets (transparently) informed about a formula φ if B get to know φ when φ is true, and B get to know the negation of φ otherwise. Transparency means that all other agents are aware of the fact that B get informed about φ, i.e., the other agents learn that $(\varphi \to C_B\varphi) \wedge (\neg\varphi \to C_B\neg\varphi)$. This action model can be defined in terms of `reveal`, as follows:

```
info :: [Agent] -> Form -> FAM
info agents form =
  reveal agents [form, negation form]
```

An example application:

```
Semantics> showM (upd e0 (info [a,b] q))
==> [0,1,2,3]
[0,1,2,3]
(0,[])(1,[p])(2,[q])(3,[p,q])
(a,[[0,1],[2,3]])
(b,[[0,1],[2,3]])
(c,[[0,1,2,3]])

Semantics> isTrue (upd e0 (info [a,b] q)) (CK [a,b] q)
False
Semantics> isTrue (upd e0 (groupM [a,b] q)) (CK [a,b] q)
True
```

Slightly different is informing a set of agents about what is actually the case with respect to formula φ:

```
infm :: EM -> [Agent] -> Form -> FAM
infm m ags f = if isTrue m f
                 then groupM ags f
                 else if isTrue m (Neg f)
                        then groupM ags (Neg f)
                        else one
```

And the corresponding thing for public announcement:

```
publ :: EM -> Form -> FAM
publ m f = if isTrue m f
             then public f
             else if isTrue m (Neg f)
                    then public (Neg f)
                    else one
```

8.4 Operations on action models

The trivial update action model is a special case of public announcement. Call this the `one` action model, for it behaves as 1 for the operation $\otimes$ of action model composition.

```
one :: FAM
one = public Top
```

Composition $\otimes$ of multiple pointed action models.

```
cmpP :: FAM -> FAM -> [Agent] -> Model (State,State) Form
cmpP fam1 fam2 ags =
 (Mo nstates npre ags nsusp npoints)
   where m@(Mo states pre _ susp ss) = fam1 ags
         (Mo states' pre' _ susp' ss') = fam2 ags
         npoints = [ (s,s') | s <- ss, s' <- ss' ]
         nstates = [ (s,s') | s <- states, s' <- states' ]
         npre    = [ ((s,s'), g) | (s,f)     <- pre,
                                   (s',f')   <- pre',
                                    g        <- [computePre m f f']  ]
         nsusp   = [ (ag,(s1,s1'),(s2,s2')) | (ag,s1,s2)    <- susp,
                                              (ag',s1',s2') <- susp',
                                               ag == ag'             ]
```

The utility function for this can be described as follows: compute the new precondition of a state pair. If the preconditions of the two states are purely propositional, we know that the updates at the states commute and that their combined precondition is the conjunction of the two preconditions, provided this conjunction is not a contradiction. If one of the states has a precondition that is not purely propositional, we have to take the epistemic effect of the update into account in the new precondition.

```
computePre  :: AM -> Form -> Form -> Form
computePre m g g'  | pureProp conj = conj
                   | otherwise     = Conj [ g, Neg (Up m (Neg g')) ]
  where conj    = canonF (Conj [g,g'])
```

Compose pairs of multiple pointed action models, and reduce the result to its simplest possible form under action emulation.

```
cmpFAM :: FAM -> FAM -> FAM
-- cmpFAM fam fam' ags = aePmod (cmpP fam fam' ags)
cmpFAM fam fam' ags = conv (cmpP fam fam' ags)
```

Use `one` as unit for composing lists of FAMs:

```
cmp :: [FAM] -> FAM
cmp = foldl cmpFAM one
```

Here is the result of composing two messages:

```
Semantics> showM (cmp [groupM [a,b] p, groupM [b,c] q] [a,b,c])
==> [0]
[0,1,2,3]
(0,&[p,q])(1,p)(2,q)(3,T)
(a,[[0,1],[2,3]])
(b,[[0],[1],[2],[3]])
(c,[[0,2],[1,3]])
```

This gives the resulting action model. Here is the result of composing the messages in the reverse order. The two action models are bisimilar under the renaming $1 \mapsto 2, 2 \mapsto 1$.

```
==> [0]
[0,1,2,3]
(0,&[p,q])(1,q)(2,p)(3,T)
(a,[[0,2],[1,3]])
(b,[[0],[1],[2],[3]])
(c,[[0,1],[2,3]])
```

The following is an illustration of an observation from [vE$_1$04a]:

```
m2  = initE [P 0,Q 0] [a,b,c]
psi = Disj[Neg(K b p),q]

Semantics>  showM (upds m2 [message a psi, message b p])
==> [1,4]
[0,1,2,3,4,5]
(0,[])(1,[p])(2,[p])(3,[q])(4,[p,q])
(5,[p,q])
(a,[[0,1,2,3,4,5]])
(b,[[0,2,3,5],[1,4]])
(c,[[0,1,2,3,4,5]])
```

```
Semantics> showM (upds m2 [message b p, message a psi])
==> [7]
[0,1,2,3,4,5,6,7,8,9,10]
(0,[])(1,[])(2,[p])(3,[p])(4,[p])
(5,[q])(6,[q])(7,[p,q])(8,[p,q])(9,[p,q])
(10,[p,q])
(a,[[0,3,5,7,9],[1,2,4,6,8,10]])
(b,[[0,1,3,4,5,6,9,10],[2,7,8]])
(c,[[0,1,2,3,4,5,6,7,8,9,10]])
```

Power of action models:

```
pow :: Int -> FAM -> FAM
pow n fam = cmp (take n (repeat fam))
```

Non-deterministic sum $\oplus$ of multiple-pointed action models:

```
ndSum' :: FAM -> FAM -> FAM
ndSum' fam1 fam2 ags = (Mo states val ags acc ss)
  where
       (Mo states1 val1 _ acc1 ss1) = fam1 ags
       (Mo states2 val2 _ acc2 ss2) = fam2 ags
       f   = \ x -> toInteger (length states1) + x
       states2' = map f states2
       val2'    = map (\ (x,y)   -> (f x, y)) val2
       acc2'    = map (\ (x,y,z) -> (x, f y, f z)) acc2
       ss       = ss1 ++ map f ss2
       states   = states1 ++ states2'
       val      = val1 ++ val2'
       acc      = acc1 ++ acc2'
```

Example action models:

```
am0 = ndSum' (test p) (test (Neg p)) [a,b,c]

am1 = ndSum' (test p) (ndSum' (test q) (test r)) [a,b,c]
```

Examples of minimization for action emulation:

```
Semantics>  showM am0
==> [0,2]
[0,1,2,3]
(0,p)(1,T)(2,-p)(3,T)
(a,[([0],[1]),([2],[3])])
(b,[([0],[1]),([2],[3])])
(c,[([0],[1]),([2],[3])])

Semantics> showM (aePmod am0)
==> [0]
[0]
(0,T)
(a,[[0]])
(b,[[0]])
(c,[[0]])
```

```
Semantics>  showM am1
==> [0,2,4]
[0,1,2,3,4,5]
(0,p)(1,T)(2,q)(3,T)(4,r)
(5,T)
(a,[([0],[1]),([2],[3]),([4],[5])])
(b,[([0],[1]),([2],[3]),([4],[5])])
(c,[([0],[1]),([2],[3]),([4],[5])])

Semantics> showM (aePmod am1)
==> [0]
[0,1]
(0,v[p,&[-p,q],&[-p,-q,r]])(1,T)
(a,[([0],[1])])
(b,[([0],[1])])
(c,[([0],[1])])
```

Non-deterministic sum $\oplus$ of multiple-pointed action models, reduced for action emulation:

```
ndSum :: FAM -> FAM -> FAM
ndSum fam1 fam2 ags = (ndSum' fam1 fam2) ags
```

Notice the difference with the definition of alternative composition of Kripke models for processes given in [Ho$_3$98, Ch 4]. The `zero` action model is the 0 for the $\oplus$ operation, so it can be used as the base case in the following list version of the $\oplus$ operation:

```
ndS :: [FAM] -> FAM
ndS = foldl ndSum zero
```

Performing a test whether φ and announcing the result:

```
testAnnounce :: Form -> FAM
testAnnounce form = ndS [ cmp [ test form, public form ],
                          cmp [ test (negation form),
                                public (negation form)] ]
```

`testAnnounce form` is equivalent to `info all_agents form`:

```
Semantics> showM (testAnnounce p [a,b,c])
==> [0,1]
[0,1]
(0,p)(1,-p)
(a,[[0],[1]])
(b,[[0],[1]])
(c,[[0],[1]])

Semantics> showM (info [a,b,c] p [a,b,c])
==> [0,1]
[0,1]
(0,p)(1,-p)
```

```
(a,[[0],[1]])
(b,[[0],[1]])
(c,[[0],[1]])
```

The function `testAnnounce` gives the special case of revelations where the alternatives are a formula and its negation, and where the result is publicly announced.

Note that *DEMO* correctly computes the result of the sequence and the sum of two contradictory propositional tests:

```
Semantics> showM (cmp [test p, test (Neg p)] [a,b,c])
==> []
[]

(a,[])
(b,[])
(c,[])

Semantics> showM (ndS [test p, test (Neg p)] [a,b,c])
==> [0]
[0]
(0,T)
(a,[[0]])
(b,[[0]])
(c,[[0]])
```

9 Examples

9.1 The riddle of the caps

Picture a situation[3] of four people a, b, c, d standing in line, with a, b, c looking to the left, and d looking to the right. a can see no-one else; b can see a; c can see a and b, and d can see no-one else. They are all wearing caps, and they cannot see their own cap. If it is common knowledge that there are two white and two black caps, then in the situation depicted in Figure 4, c knows what colour cap she is wearing.

If c now announces that she knows the colour of her cap (without revealing the colour), b can infer from this that he is wearing a white cap, for b can reason as follows: "c knows her colour, so she must see two caps of the same colour. The cap I can see is white, so my own cap must be white as well." In this situation b draws a conclusion from the fact that c knows her colour.

In the situation depicted in Figure 5, b can draw a conclusion from the fact that c does not know her colour.

In this case c announces that she does not know her colour, and b can infer from this that he is wearing a black cap, for b can reason as follows:

[3] See [vE$_1$Or05].

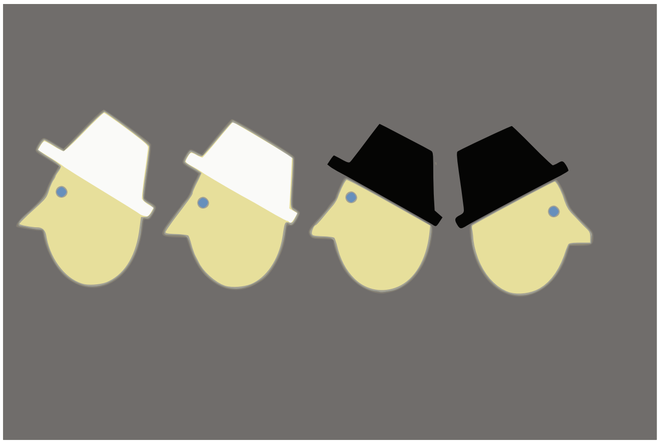


Figure 4.

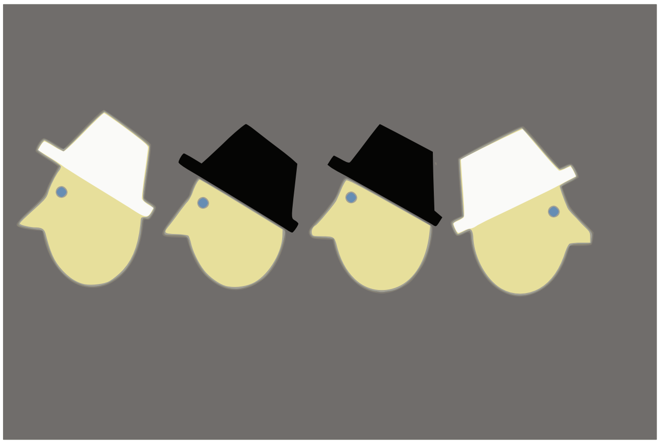


Figure 5.

"c does not know her colour, so she must see two caps of different colours in front of her. The cap I can see is white, so my own cap must be black."

To account for this kind of reasoning, we use model checking for epistemic updating, as follows. Proposition p_i expresses the fact that the ith cap, counting from the left, is white. Thus, the facts of our first example situation are given by $p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg p_4$, and those of our second example by $p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge p_4$.

Here is the *DEMO* code for this example (details to be explained below):

```
module Caps where

import DEMO

capsInfo :: Form capsInfo = Disj [Conj [f, g, Neg h, Neg j] |
                          f <- [p1, p2, p3, p4],
                          g <- [p1, p2, p3, p4] \\ [f],
                          h <- [p1, p2, p3, p4] \\ [f,g],
                          j <- [p1, p2, p3, p4] \\ [f,g,h],
                          f < g, h < j                    ]

awarenessFirstCap  = info [b,c] p1 awarenessSecondCap = info [c]
p2

cK =  Disj [K c p3, K c (Neg p3)]
bK =  Disj [K b p2, K b (Neg p2)]

mo0  = upd (initE [P 1, P 2, P 3, P 4] [a,b,c,d]) (test capsInfo)
mo1  = upd mo0 (public capsInfo)
mo2  = upds mo1 [awarenessFirstCap, awarenessSecondCap]
mo3a = upd mo2 (public cK)
mo3b = upd mo2 (public (Neg cK))
```

An initial situation with four agents a, b, c, d and four propositions p_1, p_2, p_3, p_4, with exactly two of these true, where no-one knows anything about the truth of the propositions, and everyone is aware of the ignorance of the others, is modelled like this:

```
Caps> showM mo0
==> [5,6,7,8,9,10]
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
(0,[])(1,[p1])(2,[p2])(3,[p3])(4,[p4])
(5,[p1,p2])(6,[p1,p3])(7,[p1,p4])(8,[p2,p3])(9,[p2,p4])
(10,[p3,p4])(11,[p1,p2,p3])(12,[p1,p2,p4])(13,[p1,p3,p4])
(14,[p2,p3,p4])(15,[p1,p2,p3,p4])
(a,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(b,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(c,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(d,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
```

The first line indicates that worlds $5, 6, 7, 8, 9, 10$ are compatible with the facts of the matter (the facts being that there are two white and two black caps). E.g., 5 is the world where a and b are wearing the white caps. The second line lists all the possible worlds; there are 2^4 of them, since every world has a different valuation. The third through sixth lines give the valuations of worlds. The last four lines represent the accessibility relations for the agents. All accessibilities are total relations, and they are represented here as the corresponding partitions on the set of worlds. Thus, the ignorance of the agents is reflected in the fact that for all of them all worlds are equivalent: none of the agents can tell any of them apart.

The information that two of the caps are white and two are black is expressed by the formula

$$(p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg p_4) \vee (p_1 \wedge p_3 \wedge \neg p_2 \wedge \neg p_4) \vee (p_1 \wedge p_4 \wedge \neg p_2 \wedge \neg p_3) \\ \vee (p_2 \wedge p_3 \wedge \neg p_1 \wedge \neg p_4) \vee (p_2 \wedge p_4 \wedge \neg p_1 \wedge \neg p_3) \vee (p_3 \wedge p_4 \wedge \neg p_1 \wedge \neg p_2).$$

A public announcement with this information has the following effect:

```
Caps> showM (upd mo0 (public capsInfo))
==> [0,1,2,3,4,5]
[0,1,2,3,4,5]
(0,[p1,p2])(1,[p1,p3])(2,[p1,p4])(3,[p2,p3])(4,[p2,p4])
(5,[p3,p4])
(a,[[0,1,2,3,4,5]])
(b,[[0,1,2,3,4,5]])
(c,[[0,1,2,3,4,5]])
(d,[[0,1,2,3,4,5]])
```

Let this model be called `mo1`. The representation above gives the partitions for all the agents, showing that nobody knows anything. A perhaps more familiar representation for this multi-agent Kripke model is given in Figure 6. In this picture, all worlds are connected for all agents, all worlds are compatible with the facts of the matter (indicated by the double ovals).

Next, we model the fact that (everyone is aware that) b can see the first cap and that c can see the first and the second cap, as follows:

```
Caps> showM (upds mo1 [info [b,c] p1, info [c] p2])
==> [0,1,2,3,4,5]
[0,1,2,3,4,5]
(0,[p1,p2])(1,[p1,p3])(2,[p1,p4])(3,[p2,p3])(4,[p2,p4])
(5,[p3,p4])
(a,[[0,1,2,3,4,5]])
(b,[[0,1,2],[3,4,5]])
(c,[[0],[1,2],[3,4],[5]])
(d,[[0,1,2,3,4,5]])
```

Notice that this model reveals that in case a, b wear caps of the same colour (situations 0 and 5), c knows the colour of all the caps, and in case a, b wear caps of different colours, she does not (she confuses the cases $1, 2$ and the cases $3, 4$). Figure 7 gives a picture representation.

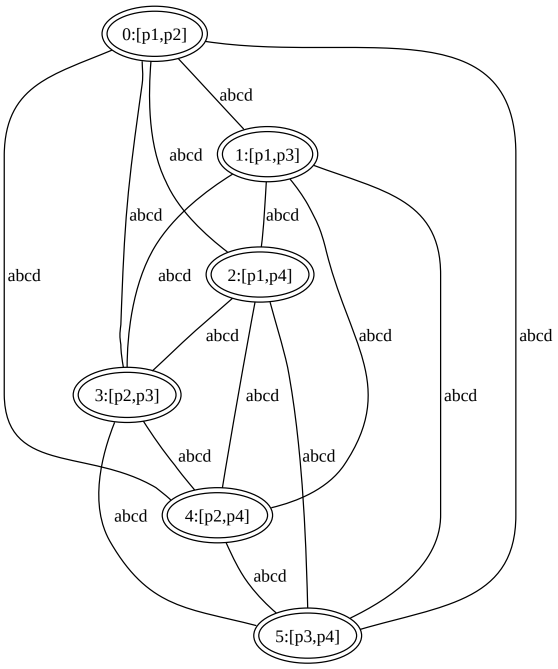


FIGURE 6. Caps situation where nobody knows anything about p_1, p_2, p_3, or p_4.

Let this model be called mo2. Knowledge of c about her situation is expressed by the epistemic formula $K_c p_3 \vee K_c \neg p_3$, ignorance of c about her situation by the negation of this formula. Knowledge of b about his situation is expresed by $K_b p_2 \vee K_b \neg p_2$. Let bK, cK express that b, c know about their situation. Then updating with public announcement of cK and with public announcement of the negation of this have different effects:

```
Caps> showM (upd mo2 (public cK))
==> [0,1]
[0,1]
(0,[p1,p2])(1,[p3,p4])
(a,[[0,1]])
(b,[[0],[1]])
(c,[[0],[1]])
(d,[[0,1]])

Caps> showM (upd mo2 (public (Neg cK)))
==> [0,1,2,3]
[0,1,2,3]
```

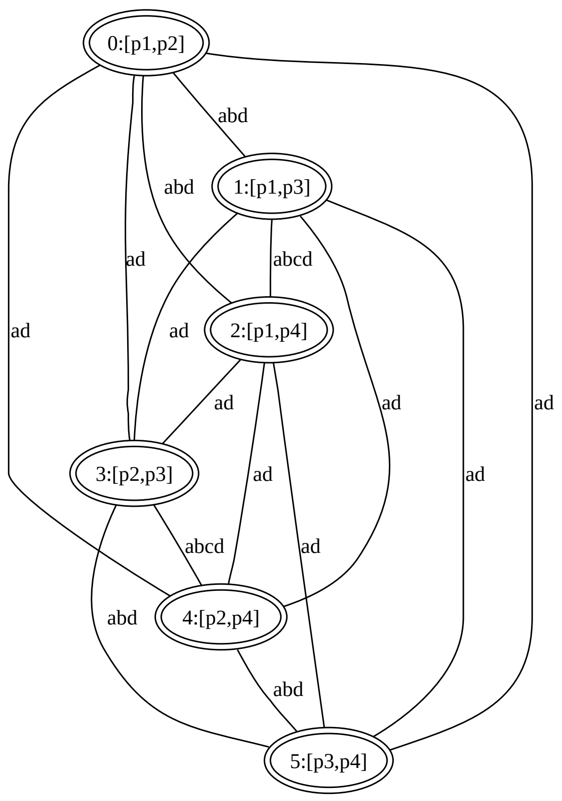


FIGURE 7. Caps situation after updating with awareness of what b and c can see.

```
(0,[p1,p3])(1,[p1,p4])(2,[p2,p3])(3,[p2,p4])
(a,[[0,1,2,3]])
(b,[[0,1],[2,3]])
(c,[[0,1],[2,3]])
(d,[[0,1,2,3]])
```

In both results, b knows about his situation, though:

```
Caps> isTrue (upd mo2 (public cK)) bK
True
Caps> isTrue (upd mo2 (public (Neg cK))) bK
True
```

9.2 Muddy children

For this example we need four agents a, b, c, d. Four children a, b, c, d are sitting in a circle. They have been playing outside, and they may or may not have mud on their foreheads. Their father announces: “At least one child is muddy!” Suppose in the actual situation, both c and d are muddy.

a	b	c	d
○	○	●	●

Then at first, nobody knows whether he is muddy or not. After public announcement of these facts, $c(d)$ can reason as follows. “Suppose I am clean. Then $d(c)$ would have known in the first round that she was dirty. But she didn't. So I am muddy.” After c, d announce that they know their state, $a(b)$ can reason as follows: “Suppose I am dirty. Then c and d would not have known in the second round that they were dirty. But they knew. So I am clean.” Note that the reasoning involves awareness about *perception*.

In the actual situation where b, c, d are dirty, we get:

a	b	c	d
○	●	●	●
?	?	?	?
?	?	?	?
?	!	!	!
!	!	!	!

Reasoning of b: “Suppose I am clean. Then c and d would have known in the second round that they are dirty. But they didn't know. So I am dirty. Similarly for c and d.” Reasoning of a: “Suppose I am dirty. Then b, c and d would not have known their situation in the third round. But they did know. So I am clean.” And so on . . . [Fa+95].

Here is the *DEMO* implementation of the second case of this example, with b, c, d dirty.

```
module Muddy where

import DEMO

bcd_dirty = Conj [Neg p1, p2, p3, p4]

awareness = [info [b,c,d] p1,
             info [a,c,d] p2,
             info [a,b,d] p3,
             info [a,b,c] p4 ]
```

```
aK =  Disj [K a p1, K a (Neg p1)]
bK =  Disj [K b p2, K b (Neg p2)]
cK =  Disj [K c p3, K c (Neg p3)]
dK =  Disj [K d p4, K d (Neg p4)]

mu0 = upd (initE [P 1, P 2, P 3, P 4] [a,b,c,d]) (test bcd_dirty)
mu1 = upds mu0 awareness
mu2 = upd  mu1 (public (Disj [p1, p2, p3, p4]))
mu3 = upd  mu2 (public (Conj[Neg aK, Neg bK, Neg cK, Neg dK]))
mu4 = upd  mu3 (public (Conj[Neg aK, Neg bK, Neg cK, Neg dK]))
mu5 = upds mu4 [public (Conj[bK, cK, dK])]
```

The initial situation, where nobody knows anything, and they are all aware of the common ignorance (say, all children have their eyes closed, and they all know this) looks like this:

```
Muddy> showM mu0
==> [14]
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
(0,[])(1,[p1])(2,[p2])(3,[p3])(4,[p4])
(5,[p1,p2])(6,[p1,p3])(7,[p1,p4])(8,[p2,p3])(9,[p2,p4])
(10,[p3,p4])(11,[p1,p2,p3])(12,[p1,p2,p4])(13,[p1,p3,p4])
(14,[p2,p3,p4])(15,[p1,p2,p3,p4])
(a,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(b,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(c,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(d,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
```

The awareness of the children about the mud on the foreheads of the others is expressed in terms of update models.

Here is the update model that expresses that b, c, d can see whether a is muddy or not:

```
Muddy> showM (info [b,c,d] p1)
==> [0,1]
[0,1]
(0,p1)(1,-p1)
(a,[[0,1]])
(b,[[0],[1]])
(c,[[0],[1]])
(d,[[0],[1]])
```

Let `awareness` be the list of update models expressing what happens when they all open their eyes and see the foreheads of the others. Then updating with this has the following result:

```
Muddy> showM (upds mu0 awareness)
==> [14]
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
```

```
(0,[])(1,[p1])(2,[p2])(3,[p3])(4,[p4])
(5,[p1,p2])(6,[p1,p3])(7,[p1,p4])(8,[p2,p3])(9,[p2,p4])
(10,[p3,p4])(11,[p1,p2,p3])(12,[p1,p2,p4])(13,[p1,p3,p4])
(14,[p2,p3,p4])(15,[p1,p2,p3,p4])
(a,[[0,1],[2,5],[3,6],[4,7],[8,11],[9,12],[10,13],[14,15]])
(b,[[0,2],[1,5],[3,8],[4,9],[6,11],[7,12],[10,14],[13,15]])
(c,[[0,3],[1,6],[2,8],[4,10],[5,11],[7,13],[9,14],[12,15]])
(d,[[0,4],[1,7],[2,9],[3,10],[5,12],[6,13],[8,14],[11,15]])
```

Call the result `mu1`. An update of `mu1` with the public announcement that at least one child is muddy gives:

```
Muddy> showM (upd mu1 (public (Disj [p1, p2, p3, p4])))
==> [13]
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]
(0,[p1])(1,[p2])(2,[p3])(3,[p4])(4,[p1,p2])
(5,[p1,p3])(6,[p1,p4])(7,[p2,p3])(8,[p2,p4])(9,[p3,p4])
(10,[p1,p2,p3])(11,[p1,p2,p4])(12,[p1,p3,p4])(13,[p2,p3,p4])
(14,[p1,p2,p3,p4])

(a,[[0],[1,4],[2,5],[3,6],[7,10],[8,11],[9,12],[13,14]])
(b,[[0,4],[1],[2,7],[3,8],[5,10],[6,11],[9,13],[12,14]])
(c,[[0,5],[1,7],[2],[3,9],[4,10],[6,12],[8,13],[11,14]])
(d,[[0,6],[1,8],[2,9],[3],[4,11],[5,12],[7,13],[10,14]])
```

Figure 8 represents this situation where the double oval indicates the actual world). Call this model `mu2`, and use `aK`, `bK`,`cK`, `dK` for the formulas expressing that a, b, c, d know whether they are muddy (see the code above). Then we get:

```
Muddy> showM (upd mu2 (public (Conj[Neg aK, Neg bK, Neg cK,
                                                     Neg dK])))
==> [9]
[0,1,2,3,4,5,6,7,8,9,10]
(0,[p1,p2])(1,[p1,p3])(2,[p1,p4])(3,[p2,p3])(4,[p2,p4])
(5,[p3,p4])(6,[p1,p2,p3])(7,[p1,p2,p4])(8,[p1,p3,p4])
(9,[p2,p3,p4])(10,[p1,p2,p3,p4])
(a,[[0],[1],[2],[3,6],[4,7],[5,8],[9,10]])
(b,[[0],[1,6],[2,7],[3],[4],[5,9],[8,10]])
(c,[[0,6],[1],[2,8],[3],[4,9],[5],[7,10]])
(d,[[0,7],[1,8],[2],[3,9],[4],[5],[6,10]])
```

This situation is represented in Figure 9. We call this model `mu3`, and update again with the same public announcement of general ignorance:

```
Muddy> showM (upd mu3 (public (Conj[Neg aK, Neg bK, Neg cK,
                                                     Neg dK])))
==> [3]
[0,1,2,3,4]
(0,[p1,p2,p3])(1,[p1,p2,p4])(2,[p1,p3,p4])(3,[p2,p3,p4])
(4,[p1,p2,p3,p4])
```

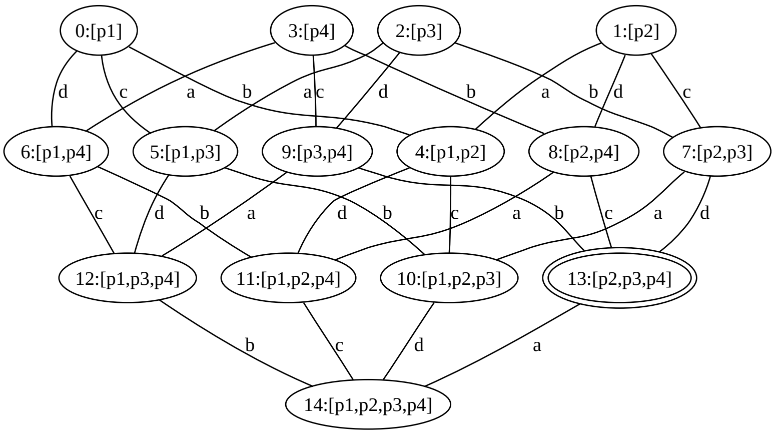


FIGURE 8.

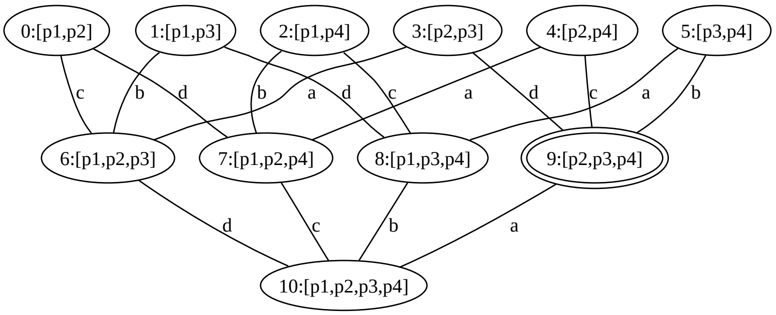


FIGURE 9.

```
(a,[[0],[1],[2],[3,4]])
(b,[[0],[1],[2,4],[3]])
(c,[[0],[1,4],[2],[3]])
(d,[[0,4],[1],[2],[3]])
```

Finally, this situation is represented in Figure 10, and the model is called `mu4`. In this model, b, c, d know about their situation:

```
Muddy> isTrue mu4 (Conj [bK, cK, dK])
True
```

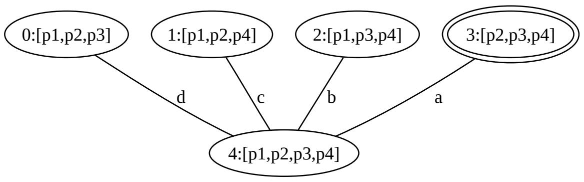


FIGURE 10.

Updating with the public announcement of this information determines everything:

```
Muddy> showM (upd mu4 (public (Conj[bK, cK, dK])))
==> [0]
[0]
(0,[p2,p3,p4])
(a,[[0]])
(b,[[0]])
(c,[[0]])
(d,[[0]])
```

10 Conclusion and further work

DEMO was used for solving Hans Freudenthal's Sum and Product puzzle by means of epistemic modelling in [vDRu$_0$Ve$_2$05]. There are many variations of this. See the *DEMO* documentation at `http://www.cwi.nl/~jve/demo/` for descriptions and for *DEMO* solutions. *DEMO* is also good at modelling the kind of card problems described in [vD03], such as the Russian card problem. A *DEMO* solution to this was published in [vD+06]. *DEMO* was used for checking a version of the Dining Cryptographers protocol [Ch$_2$88], in [vE$_1$Or05]. All of these examples are part of the *DEMO* documentation.

The next step is to employ *DEMO* for more realistic examples, such as checking security properties of communication protocols. To develop *DEMO* into a tool for blackbox cryptographic analysis — where the cryptographic primitives such as one-way functions, nonces, public and private key encryption are taken as given. For this, a propositional base language is not sufficient. We should be able to express that an agent A generates a nonce n_A, and that no-one else knows the value of the nonce, without falling victim to a combinatorial explosion. If nonces are 10-digit numbers then not knowing a particular nonce means being confused between 10^{10} different worlds. Clearly, it does not make sense to represent all of these in an implementation. What could be done, however, is represent epistemic models as triples (W, R, V), where V now assigns a non-contradictory proposition to each world. Then uncertainty about the value of n_A, where the actual value is N, can be represented by means of two worlds, one where $n_a = N$ and one where $n_a \neq N$. This could be done with basic propositions of the form $e = M$ and $e \neq M$, where e ranges over cryptographic expressions, and M ranges over 'big numerals'. Implementing these ideas, and putting *DEMO* to the test of analysing real-life examples is planned as future work.

References

[Ba$_4$02] A. Baltag. A logic for suspicious players: epistemic action and belief-updates in games. *Bulletin of Economic Research* 54(1):1–45, 2002.

[Ba$_4$Mo$_3$So$_1$99] A. Baltag, L. Moss & S. Solecki. The logic of public announcements, common knowledge, and private suspicions. SEN-R9922, CWI, Amsterdam, 1999.

[Ba$_4$Mo$_3$So$_1$03] A. Baltag, L. Moss & S. Solecki. The logic of public announcements, common knowledge, and private suspicions. Dept of Cognitive Science, Indiana University and Dept of Computing, Oxford University, 2003.

[BldRVe$_1$01] P. Blackburn, M. de Rijke & Y. Venema. *Modal Logic*, *Cambridge Tracts in Theoretical Computer Science* 53. Cambridge University Press, 2001.

[Ch$_1$Ko$_0$Po$_0$06] Z. Chatzidakis, P. Koepke & W. Pohlers, eds. *Logic Colloquium '02*, *Lecture Notes in Logic* 27. Association for Symbolic Logic, 2006.

[Ch$_2$88] D. Chaum. The dining cryptographers problem: unconditional sender and receiver untraceability. *Journal of Cryptology* 1(1):65–75, 1988.

[Ch$_3$80] B. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, 1980.

[Da$_1$Lo$_0$Lo$_4$62] M. Davis, G. Logemann & D. Loveland. A machine program for theorem proving. *Communications of the ACM* 5(7):394–397, 1962.

[Da$_1$Pu60] M. Davis & H. Putnam. A computing procedure for quantification theory. *Journal of the ACM* 7(3):201–215, 1960.

[Fa+95] R. Fagin, J. Halpern, Y. Moses & M. Vardi. *Reasoning about Knowledge.* The MIT Press, 1995.

[Ga$_0$Ko$_5$No$_0$06] E. Gansner, E. Koutsofios & S. North. *Drawing graphs with* dot. Graphviz project, 2006. URL `http://www.graphviz.org`.

[Ge$_2$99a] J. Gerbrandy. *Bisimulations on planet Kripke.* Ph.D. thesis, Universiteit van Amsterdam, 1999. *ILLC Publications* DS-1999-01.

[Go$_0$02] R. Goldblatt. *Logics of Time and Computation, Second Edition, Revised and Expanded*, *CSLI Lecture Notes* 7. CSLI, Stanford, 1992.

[Hi$_1$62] J. Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions.* Cornell University Press, 1962.

[Ho$_3$98] M. Hollenberg. *Logic and Bisimulation.* Ph.D. thesis, Utrecht University, 1998.

[Ho$_4$71] J.E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In [Ko$_1$Pa$_8$71, pp. 189–196].

[Jo$_2$03] S.P. Jones. *Haskell 98. Language and Libraries. The Revised Report.* Cambridge University Press, 2003.

[Ko$_1$Pa$_8$71] Z. Kohavi & A. Paz, eds. *Theory of Machines and Computations.* Academic Press, 1971.

[Ko$_4$03] B. Kooi. *Knowledge, Chance and Change.* Ph.D. thesis, University of Groningen, 2003. *ILLC Publications* DS-2003-01.

[Mo$_3$Gi$_1$dR99] L.S. Moss, J. Ginzburg & M. de Rijke, eds. *Logic, Language and Information*, 2. CSLI Publications, 1999.

[$\text{Pa}_1\text{Ta}_0$87] R. Paige & R.E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing* 16(6):973–989, 1987.

[$\text{Ru}_0$04] J. Ruan. *Exploring the update universe.* Master's thesis, Universiteit van Amsterdam, 2004. *ILLC Publications* MoL-2004-08.

[vB01b] J. van Benthem. Language, logic, and communication. In [vB+01, pp. 7–25].

[vB06] J. van Benthem. 'One is a Lonely Number': On the logic of communication. In [$\text{Ch}_1\text{Ko}_0\text{Po}_0$06, pp. 96–129].

[vB+01] J. van Benthem, P. Dekker, J. van Eijck, M. de Rijke & Y. Venema, eds. *Logic in Action.* ILLC, 2001.

[vD00] H. van Ditmarsch. *Knowledge Games.* Ph.D. thesis, University of Groningen, 2000. *ILLC Publications* DS-2000-06.

[vD03] H. van Ditmarsch. The Russian cards problem. *Studia Logica* 75(1):31–62, 2003.

[$\text{vDRu}_0\text{Ve}_2$05] H. van Ditmarsch, J. Ruan & R. Verbrugge. Model checking sum and product. In [$\text{Zh}_1\text{Ja}_3$05, pp. 790–795].

[vD+06] H. van Ditmarsch, W. van der Hoek, R. van der Meyden & J. Ruan. Model checking Russian cards. *Electronic Notes in Theoretical Computer Science* 149(2):105–123, 2006.

[$\text{vE}_0$05] M. van Eekelen, ed. *6th Symposium on Trends in Functional Programming, TFP 2005: Proceedings.* Institute of Cybernetics, Tallinn, 2005.

[$\text{vE}_1$04a] J. van Eijck. Communicative actions, 2004. Preprint.

[$\text{vE}_1$04b] J. van Eijck. Reducing dynamic epistemic logic to PDL by program transformation. SEN-E0423, CWI, Amsterdam, December 2004.

[vE_1Or05] J. van Eijck & S. Orzan. Modelling the epistemics of communication with functional programming. In [$\text{vE}_0$05, pp. 44–59].

[$\text{Zh}_0\text{St}_5$00] H. Zhang & M.E. Stickel. Implementing the Davis-Putnam method. *Journal of Automated Reasoning* 24(1–2):277–296, 2000.

[$\mathrm{Zh_1Ja_305}$] S. Zhang & R. Jarvis, eds. *AI 2005: Advances in Artificial Intelligence, 18th Australian Joint Conference on Artificial Intelligence, Sydney, Australia, December 5–9, 2005, Proceedings*, *Lecture Notes in Computer Science* 3809. Springer, 2005.