

Combinatory Linguistics

Combinatory Linguistics

by

Cem Bozşahin

De Gruyter Mouton

ISBN 978-3-11-025170-8
e-ISBN 978-3-11-029687-7

Library of Congress Cataloging-in-Publication Data

A CIP catalog record for this book has been applied for at the Library of Congress.

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <http://dnb.dnb.de>.

© 2012 Walter de Gruyter GmbH, Berlin/Boston

Printing: Hubert & Co. GmbH & Co. KG, Göttingen

⊗ Printed on acid-free paper

Printed in Germany

www.degruyter.com

in memory of:

Dođan Bozřahin
Ferhunde Bozřahin
Saliha İdemen
Ferruh İdemen

Preface

This book is about the place and role of combinators in linguistics, and through it, in cognitive science, computational linguistics and philosophy. It traces the history of Combinatory Categorical Grammar (CCG) and presents its linguistic implications. It aims to show how combinatory theories and models can be built, evaluated and situated in the realm of the four fields. The introductory remarks in the beginnings of early chapters can hopefully be excused because of the wide target readership.

The book examines to what extent knowledge of words can be construed as the knowledge of language, and what that knowledge might look like, at least on paper. It studies the semantic mechanism that engenders directly interpretable constituents, the combinators, and their limits in a grammar. More specifically, it investigates the mediating relation between constituents and their semantics insofar as it arises from combinatory knowledge of words and syntacticized combinators. It is not about forms or meanings *per se*.

Its key aspect is to promote the following question as a basic scientific inquiry of language: why do we see limited dependency and constituency in natural language syntax? We owe the question to David Hume by a series of links, some of which are covered in the book. The reader might be puzzled by this claim, knowing that Hume had said very little about language. I believe he avoided it for a good reason, but the question goes back to him nevertheless, as I try to argue in the book.

It seems that thinking syntax is syntax and semantics is semantics in their own structure isn't going to take us too far from the knowledge we have accumulated on grammars, about what they can and cannot do regarding code-terminism in forms and meanings, and about the coconspiracy of forms and meanings. The same goes, I am sure, to thinking discourse is discourse, morphology is morphology etc. The book focuses on the relationship between syntax and semantics.

Many explanans about syntactic processes become explananda when we readjust our semantic radar, a term which I use as a metaphor for looking at semantic objects with a syntactic eye. As all metaphors are, it is somewhat misleading in the beginning, which I hope becomes less of a metaphor as we proceed. If we open the radar too wide, we are forced to do syntax with semantic types, and run the risk of missing the intricate and complex syntac-

tic dependencies, which in turn might miss an opportunity to limit “possible human languages”. If it is too narrow, we must do semantics with syntactic types, and that might take us to the point of having syntaxes rather than syntax. Both extremes need auxiliary assumptions to provide a constrained theory of language.

Many syntactic dependencies turn out to be semantic in nature, and these dependencies seem to arise from a single resource. This resource is conjectured to be adjacency. The conjecture of semantics arising from order goes back about a century in mathematics, to Schönfinkel, and almost half a century in philosophy, linguistics and cognitive science, to Geach, Ades and Steedman. The natural meeting point of the two historically independently motivated theorizing about adjacency, the semantic and the syntactic one about combinators, is the main story of the book.

In this regard, the book was bound to be a historical account from the beginning. However, it came to provide, in some detail, ways of theory and model construction for linguistics and cognitive science in which there is no degree of freedom from adjacency. This pertinacious course seems to set up the crucial link between forms and meanings with as little auxiliary assumptions as its progenitors can think of. I believe it sets up creative links in theorizing about the computational, linguistic, cognitive and philosophical aspects of grammar. I exemplify these connections one by one.

When we look at combinators as functions they are too powerful, equivalent to the power of a Turing machine. As such they cannot do linguistic work because natural language constituency narrows down the expressible semantic dependencies manifested by functions. The linguistic theorizing begins when we syntacticize the combinators and establish some criteria about which combinator must be in the grammar and which one can materialize in the lexicon. An explanatory force can be reached if the process reveals predictions about possible constituents, possible grammars and possible lexicons, without the need for variables and within a limited computational power. Structure-dependence of natural language strings can be predicted too, rather than assumed.

Every intermediate constituent will be immediately interpretable, and non-constituents will be uninterpretable by this process. In other words, being a constituent, being derivable and being immediately interpretable are three manifestations of the same property: natural grammars are combinatory type-dependent. These are the narrow claims of Combinatory Categorical Grammar.

The notion of grammar is trivialized if there is no semantics in it. Some, like Montague, went as far as claiming that syntax is only a preliminary for semantics. On the other hand, language would not be worth theorizing about if the semantics we have in mind for grammars is the semantics out there. All species do this all the time without language, to mean things as they see fit. Words would be very unnecessary, as one songwriter put it in the early 90's.¹ Perhaps they are not always there, as in the lyrics of Elizabeth Fraser.² Sadly, words are needed for us mortals, and somewhat surprisingly, they are more or less sufficient, if we take them as personal interrelated histories of what connects the dots in syntax and compositional semantics, which is embodied in their syntactic combinatory type, as knowledge arising more than the experience. Herein lies a Humean story.

Although I have tried to keep it to a minimum to compare the present theory with others, for the sake of brevity and focus, the historical perspective in the book makes unavoidable points of contact with different ways of theorizing about grammars. Some examples are worth noting from the beginning. (a) Steedman's and Jacobson's use of combinators for syntax differs when it comes to reference and quantifier scope. (b) Kayne claims that structure determines order, with directionally-constrained syntactic movement being the key element in explanations. Order determines structure in the combinatory theory, and no-movement is the key to explanations. (c) HPSG is another type-dependent theory of syntax like the one presented in the book. HPSG's types are related to each other by subtyping, whose semantics do not necessarily arise from order. (d) Type-logical grammar in particular and Montague's use of type-theoretic language in general use semantic types to give rise to meaningful expressions, that is, to syntax. Order is not necessary or sufficient for a set-based type's construal, therefore it need not be the basis for meaningful expressions. (e) Obviously not all categorial grammars are combinatory categorial grammars. The telltale signs of the latter kind, which is the main topic of the book, are no use of phonologically null types, no use of surface wrap, some use of type combination that goes above function application, and the insistence on an order-induced syntax-semantics for every rule and lexical category, as opposed to for example order and structural unification. (f) Dependency grammars take dependency as an asymmetric relation of words in a string, i.e. as a semantic relation between syntactic objects, but leave open why there are limited kinds of dependencies, and why these dependencies relate to surface constituency and interpretability in predictable ways. (g) Chomsky's program can be seen as a concerted effort to squeeze as

much compositional semantics into syntax as possible. The A-over-A principle, the X-bar model, subadjacency, cyclicity, filters, functional categories, main thematic condition, chains, crash and the process of derivation-by-phase do to syntactic trees what they cannot do by themselves: constrain the possible semantic interpretations of the syntactic objects in them hypergrammatically.

The apparent similarities of these theories must be put in context. As Pollard points out frequently, most theories subscribe to some form of syntactocentrism because they conceive the relation between forms and meanings as indirect. It must be mediated by syntax. The theory covered in the book is syntactocentric in Pollard's sense. The syntactocentrism that will be argued against here is the one that sees semantics as an appendix to syntax. The theory presented here is neither the first nor the only remaining one on this stance.

We need only look thirtysomething years before the rise of that kind syntactocentrism to find an alternative foundation for bringing semantics into syntax. Two historically independent programs, radical lexicalization and codeterminacy of syntax and semantics, culminate in a theory where adjacency is the only fundamental assumption. Two aspects will figure prominently: dependency and constituency. Both will get their interpretation from a single source, the semantics of order.

For the reader: the book is organized in such a way that the technical material that gets in the way of linguistics has been moved to appendices. This leaves some aspects of combinators, grammars and computing to the appendices (mostly definitions and basic techniques). Linguistic theorizing about the combinators is in the main text. There is no reference to the appendices from the main matter, or from appendices to the chapters in the main body. The back matter might refer to earlier ones. Reading all the appendices in the given order might help readers who are unfamiliar with some of the terminology.

Now to pay some debts old and new academic and personal. This book started as my I-see-what-they-mean project, although I am not sure about the end result. It was an attempt at a collective understanding of Moses Schönfinkel, Mark Steedman, Anna Szabolcsi, Pauline Jacobson, Noam Chomsky, Richard Montague, Haskell Curry, Emmon Bach and John Robert Ross, among others. I hope the reader does not visit my shortcomings on them.

At a more personal level, my first contact with Mark and his theory was in the years 1992–1994, and since then it has become a major part of my academic life. I have asked so many questions to Mark that I am slightly

embarrassed I am getting away with an acknowledgment. Before then I was fortunate to be taught by great teachers, whom I'm honored to list in somewhat chronological order: Türkân Barkın, Metin Ünver, İbrahim Nişancı, late Esen Özkarahan, Nicholas Findler and Leonard 'Aryeh' Faltz. Some friends and family taught me more on academic affairs than I was able to acknowledge so far. There is a bit of them in the book but I cannot exactly point where. Thank you Canuş, *née* Cihan Bozşahin, Neziha Aytaçlar, Zafer Aracagök, Uğur Atak, Ragıp Gürkan, Justin Coven, Uttam Sengupta, Halit Oğuztüzün, Samet Bağçe, Sevil Kıvan, Aynur Demirdirek, Stasinou Konstantopoulos, Mark McConville, Harry Halpin, İrem Aktuğ, Mark Ellison and Stuart Allardyce.

Mark Steedman, Ash Asudeh and Frederick Hoyt provided comments on much earlier drafts. Umut Özge was less fortunate to have gone through several drafts. I owe some sections to discussions with him, and with Ceyhan Temürcü, Mark Steedman and Aravind Joshi. Elif Gök, Yağmur Sağ, Süleyman Taşçı, Deniz (*Dee!*) Zeyrek and Alan Libert suggested corrections and clarifications for which I am grateful. Finally, special thanks to the Mouton team, Uri Tadmor, Birgit Sievert, Julie Miess, Angelika Hermann and the reviewers for comments and assistance with the manuscript. Livia Kortvelyessy of Versita helped me get the project going.

I am solely responsible for not heeding good advice of so many good people.

Contents

List of Tables	xvii
1 Introduction	1
2 Order as constituent constructor	9
1 Combinatory syntactic types	9
2 Directionality in grammar: morphology, phonology or syntax?	11
3 Trees and algorithms	13
4 CCG’s narrow claims in brief	16
5 Type-dependence versus structure-dependence	17
6 Constituency	22
3 The lexicon, argumenthood and combinators	31
1 Adjacency and arity	32
2 Words, supercombinators and subcombinators	33
3 Infinitude and learnability-in-principle	36
4 Syntacticizing the combinators	43
1 Unary combinators	45
2 Binary combinators	47
3 Ternary combinators	49
4 Quaternary combinators	52
5 Powers and combinations	55
6 Why syntacticize?	58
5 Combinatory Categorical Grammar	61
1 Combinators and wrapping	61
2 Linguistic categories	65
3 CCG is nearly context-free	73
4 Invariants of natural language combination	74
5 The BTS system	82
6 The LF debate	87
1 Steedman’s LF	89

2	Szabolcsi's reflexives	92
3	Jacobson's pronouns	94
4	More on LF: Unary BCWZ , constituency and coordination . .	100
7	Further constraints on possible grammars	107
8	A BTSO system	113
9	The semantic radar	121
1	Boundedness and unboundedness	122
2	Recursive thoughts and recursive expressions	132
3	Grammar, lexicon and the interfaces	137
4	Making CCG's way through the Dutch impersonal passive . .	142
5	Computationalism and language acquisition	149
6	Stumbling on to knowledge of words	156
7	Functional categories	163
8	Case, agreement and expletives	170
9	The semantics of scrambling	173
10	Searle and semantics	177
10	Monadic computation by CCG	183
1	Application	184
2	Dependency	190
3	Sequencers	192
4	The CCG monad	194
5	Radical lexicalization revisited	198
6	Monadic results and CCG	200
11	Conclusion	205
	Appendices	215
	Appendix A: Lambda calculus	217
	Appendix B: Combinators	219
	Appendix C: Variable elimination	223
	Appendix D: Theory of computing	225

Appendix E: Radical lexicalization and syntactic types	229
Appendix F: Dependency structures	233
Notes	235
Bibliography	249
Author and name index	272
Subject index	278

List of Tables

1	Basic combinators	45
2	The syntacticized BTSO system	117
3	Tad's first words	150
4	Keren's first words	151
5	Growth rates of polynomial and exponential functions	155
6	Phonology-driven encoding of monadic dependencies	193
7	Some well-known combinators	220

Chapter 1

Introduction

On December 7, 1920, Moses Ilyich Schönfinkel made mathematical history when he presented to the Göttingen Mathematical Society his results about variables. It was to be his only work on the topic, which was prepared for publication by Behmann (Schönfinkel 1920/1924).³ Little would he know that in this brief seminar he was going to change the course of computing and linguistics too, two fields which flourished in the remainder of the century.⁴

He simply eliminated variables—bound variables. In theory, any lambda term with no free variables is a combinator in Schönfinkel’s sense, and all the bound variables in it can be eliminated. In practice, two combinators suffice to compute any discretely representable dependency, and that takes us to language and computing. We shall see that although this is good news for computing because we can rigorously identify a computable fragment of functions, it requires much extra effort in linguistics to become a theory because we will need some empirical criteria and a theory to constrain this power: we know that human languages do not manifest every computable dependency.

The list of names who worked on the variable reads as a “who is who” in mathematics and philosophy: Curry, Frege, Herbrand, Hilbert, Peirce, Rosser, Skolem, and later, Quine and de Bruijn. They were a concern for the mathematician, linguist, logician, philosopher, and the computer scientist. Naturally, discovery of different methods was expected.⁵

The way Schönfinkel set about to go at it is what made the overarching influence beyond mathematics. He gave semantics to order, and order alone, by devising an ingenious way to represent all argument-taking objects uniformly, something that eluded Frege in his lifetime although he had anticipated it.

Schönfinkel represented a function f of n arguments (1a) as an n -sequence of one-argument functions (1b). Assuming left-associativity for juxtaposition, it is now standard practice to write (1b) as (1c), as Schönfinkel did.

- (1) a. $f(x_1, x_2, \dots, x_n)$
b. $(\dots((fx_1)x_2)\dots x_n)$
c. $fx_1x_2 \dots x_n$

2 Introduction

This is what we now call *Currying*. (I must confess that *Schönfinkeling* is alive in secret sects.) Haskell Brooks Curry was the first to realize the importance of the technique, and made very frequent use of it in his arguments (and Schönfinkel faded into oblivion), hence the name. The technique had been taken for granted mainly because it was so simple.

Its manifestation in language can be easily taken for granted too. It translates to one-at-a-time word-taking in a surface string such as (2).

(2) ((((((*I wonder*) *who*) *Kafka*) *might*) *have*) *liked*.)

Every parenthesized expression in the example except the outermost one is syntactically incomplete, yet semantically interpretable, in the sense of being a function of type $\lambda x.m'/x$, where m' is a crude way to symbolize the incrementally assembled semantics of each parenthesis.

For an n -argument object $f(x_1, \dots, x_n)$ written in the traditional notation, we can obtain a variableless representation of f using eta-reduction and combinators. Variable-free functions capture the content and its combinatory behavior without reference to extraneous objects.

An early comparison of variable-friendly syntax with variable-free syntax shows us that the aim is not to simply clean up theoretical tools and vocabulary. Its primary motivation is empirical: if a string of objects can have a variable-free function, then they are immediately interpretable. Taken to its logical conclusion, it means that any intermediate phrase has instantly available semantics. For example, *the man who Mary loved* is taken to arise from the structure *the man who* [*Mary loved* ___] in variable-friendly syntax, where the empty element (a syntactic variable) awaits interpretation. Consequently, the phrase it is part of waits for interpretation too. In variable-free syntax, *Mary loved* is semantically $\lambda x.\mathbf{C}love'mary'/x$, which is eta-convertible to $\mathbf{C}love'mary'$, where \mathbf{C} is one of Curry's combinators. It does not need anything else to be interpretable. It needs something to become a proposition, but that is more than being interpretable. By direct import of combinators to variable-free syntax, we get immediately interpretable intermediate constituents as well. This is the main story of the book.

Variables cannot be eliminated at the expense of lexical proliferation or loss of semantics. For example, we cannot assume that *love* above is intransitive, which would give the structure [*Mary loved*]. That is to say that all strings are inherently *typed* as grammatical objects, such as the word *loved* (transitive) and its meaning *love'*, which is $(e, (e, t))$. Here I follow the tradition of writing the meaning of words with primes. The ubiquitous adage *The*

meaning of life is life,' attributed by Carlson (1977) to Barbara Partee and Terry Parsons, will serve as a convenient base for compositional semantics in subsequent chapters.

For us to continue giving semantics to any intermediate phrase, the argument structure of words must be carried too. We can take $\lambda x \lambda y. \text{mark}'xy$ to be equivalent to *mark,*' as in *Twain marks two fathoms*. Such carried abstractions are required by phonology because we cannot substitute two or more arguments at the same time.

We would be home and dry if all function-argument dependencies in language were that simple, but we know that for example $\lambda x. fxx$ and $\lambda x. fx(gx)$ are possible configurations, hence simple eta-conversion would not always work. Some examples are *Kinski adored himself*, which has the dependencies *adore'**kinski'**kinski,*' and *I have read without understanding*, which is *(not'understand' x)(read' xi')i'*, for some *x*, for example *the books I have read without understanding*.

The problem of capturing dependency, constituency and immediate interpretation is exacerbated by mixed-branching (3a) and right-branching (3b) demanded by language:

- (3) a. *(I wonder) (who Kafka might have liked) (and what Wittgenstein might have written.)*
 b. *I (begin (to (try (to (avoid (reading Kafka before sleep))))))*

The informal notion of constituency I employ here and denote with parentheses will be clarified throughout the book, which is inextricably tied to dependency, intonation, informativity and interpretability, and by direct import of combinators, to syntactic combinability.

Notice the tension between left-to-right carried open interpretations such as (4a) and the rightward dependencies required by semantics, as in (3b). Both kinds of branching are reflected in syntax by constituency, for example (4b).

- (4) a. *((((((((I begin) to) try) to) avoid) reading) Kafka) before) sleep.)*
 b. *(I begin to try to avoid), (and you should refrain from), (reading Kafka before sleep.)*

The inadequacy of eta-conversion for the semantic side of the constituents is where Schönfinkel's combinators come into the picture. For example, the dependency in $\lambda x. fxx$ is not eta-reducible to *f*, hence we have no way of capturing the dependencies in *Kinski adored himself* without variables or combinators. We can eta-normalize $\lambda x. fxx$ to $\lambda x. \mathbf{W}fx =_{\eta} \mathbf{W}f$, without variables.

4 Introduction

We can say that **BSC** symbolizes the dependency $\lambda x.f(gx)x$ which we can observe in the bracketed part of the string *the books_x [I have read _x without understanding _x]*, without variables.

$$(5) \mathbf{BSC}(\text{not}'\text{understand}')(\text{read}')i' = (\text{not}'\text{understand}')(\text{read}'i')i'$$

We can also assume that the inner dependency symbolized by the syntactic variable '_x' is **S**:

$$(6) \mathbf{S}(\text{not}'\text{understand}')\text{read}' = \lambda x.(\text{not}'\text{understand}'x)(\text{read}'x)$$

Given the combinators and the process of eta-normalization, knowing a word in the combinatory sense becomes the problem of capturing its predicate-argument dependency structure in direct correspondence with its syntax and constituent structure, without variables.

Schönfinkel's method allows us to capture the syntacticization of semantic dependencies with a handful of combinators, all of which are based on adjacency. Below is the semantic side of the story, where the strings in parentheses are interpreted.⁶

$$(7) \text{ a. } (\text{Kafka adored}) \text{ and Wittgenstein loathed mentors.}$$

$$\mathbf{B}(\mathbf{T}\text{kafka}')\text{adore}' = \lambda x.\text{adore}'x\text{kafka}'$$

$$\text{ b. } I \text{ offered, and (may give), a flower to a policeman. (Steedman 1988)}$$

$$\mathbf{B}^2\text{may}'\text{give}' = \lambda x\lambda y\lambda z.\text{may}'(\text{give}'xyz)$$

$$\text{ c. } He \text{ is the man I will (persuade every friend of) (to vote for). (Steedman 1996b)}$$

$$\mathbf{S}\text{pefo}'\text{tvf}' = \lambda x.\text{pefo}'x(\text{tvf}'x)$$

$$\text{ d. } (\text{What you can}) \text{ and what you must not base your verdict on (Hoyt and Baldrige 2008)}$$

$$\mathbf{O}(\lambda Q.?\text{x}Qx)(\text{you}'\text{can}') = ?x\lambda P.\text{can}'(Px\text{you}')$$

The combinators involved in (7) are all that we need for human languages. (And they have a common bond; see the conclusion.) This is the conjecture of CCG. The book attempts to show how CCG builds these dependency and constituency structures through syntactic types. It pairs phonological strings with predicate-argument structures in a radically lexicalized manner.

Here is the preview of the syntax of these constituents. We shall see how the semantically-motivated combinators above lead to the syntactically-realized ones below, by a direct translation made possible by the semantics of order. We will need a linguistic theory in addition to this translation be-

cause the claim is that not all the combinators can materialize as syntactic.

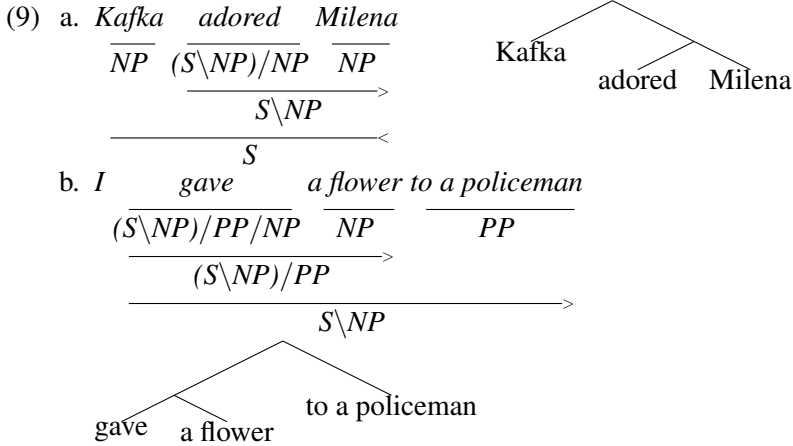
- (8) a.
$$\frac{\frac{\overline{Kafka} \quad \overline{adored}}{NP \quad (S \setminus NP) / NP}}{S / (S \setminus NP)} \xrightarrow{\mathbf{T}}$$

$$\xrightarrow{\mathbf{B}}$$
- b.
$$\frac{\overline{may} \quad \overline{give}}{(S \setminus NP) / (S \setminus NP) \quad (S \setminus NP) / PP / NP} \xrightarrow{\mathbf{B}^2}$$
- c.
$$\frac{\overline{persuade \ every \ friend \ of} \quad \overline{to \ vote \ for}}{(S \setminus NP) / VP / NP \quad VP / NP} \xrightarrow{\mathbf{S}}$$
- d.
$$\frac{\overline{What} \quad \overline{you \ can}}{S / (S \setminus NP) \quad S / VP} \xrightarrow{\mathbf{O}}$$

$$S / (VP / NP)$$

These examples also show the workings of a syntactic type-driven derivation. The syntactic types of the meaning-bearing elements do all the work in derivations. By a common convention dating back to 1930s (Ajdukiewicz 1935), the derivations are shown bottom-up, with leaves on top and the root at the bottom. Each line is a step of the derivation. Unlike phrase-structure trees which show a description of structure, these sequences are algorithms of structure-building by the string. The string span of a derivation shows the coverage of the substring for the derivation. The combinator that engenders the derivation is written at the right edge for exposition. In these example it is the syntacticized version of **BTSO**, decorated as e.g. ($\mathbf{>B}$). Semantic assembly is immediate (and not always shown), precisely because of the combinatory source of every syntacticized combinator.

The syntactic types of (8) are related to semantic types of (7) systematically. For example, (7a) suggests that *Kafka'* is type-raised by **T**, which manifests itself as the syntactic type $S / (S \setminus NP)$ in English (8a). It undergoes **B** with *adore'* semantically according to (7a), which materializes syntactically as the composition of $S / (S \setminus NP)$ and $(S \setminus NP) / NP$, which is an instance of syntactic **B**, $X / Y \ Y / Z \rightarrow X / Z$. Traditional constituents which are familiar from tree drawings, such as those in (9), will turn out to be a consequence of the combinatory primitive, function application, decorated as ($\mathbf{>}$) and ($\mathbf{<}$).



It would be tempting to think of slash introduction (directionality) on the syntactic side as the equivalent of eta-conversion on the semantic side, but that would be misleading. If it were true, we could do syntax completely with semantic types. The syntactic type of the word *adore* above is indeed equivalent to its eta-normalizable semantics $\lambda x \lambda y. \text{adore}'xy$, i.e. one slash per lambda-binding, but these slashes depend on surface adjacency, hence e.g. $(S/NP)/NP$ would be wrong for *adore* or for any English transitive verb. Additionally, some lambdas are not syntactic lambdas, e.g. $\lambda x. \text{man}'x$ for the word *man*, which is eta-normalizable to *man'* but its syntax is not N/N or $N \setminus N$ in English. These aspects show that combinators and their one-to-one syntacticization do not amount to a linguistic theory. This is where the linguistic theorizing begins for combinators.

Let me finish the preliminaries of the book with an assessment of Schönfinkel by Quine. I shall return to this quote in the final chapter.

It was letting functions admit functions generally as arguments that Schönfinkel was able to transcend the bounds of the algebra of classes and relations and so to account completely for quantifiers and their variables, as could not be done within that algebra. The same expedient carried him, we see, far beyond the bounds of quantification theory in turn: all set theory was his province. His C, S, U and application are a marvel of compact power. But a consequence is that the analysis of the variable, so important a result of Schönfinkel's construction, remains all bound up with the perplexities of set theory. Quine (1967: 357)

The essence of combinators for language is to turn a simple concept like adjacency into a scientific tool with clear limits and predictable syntactic and se-

mantic (im)possibilities, precisely because variables are eliminated to model adjacency or adjacency-like effects. And without them constituency and dependency can easily tell whether our hypothesis about a certain construction is right or wrong. That seems desirable for achieving descriptive adequacy of grammars. There is very little degree of freedom when the entire theory is based on a single understanding of adjacency. That will hopefully carry an explanatory force when syntax and semantics are considered together.

The rest of the book is organized as follows. Chapter 2 introduces type-dependent syntax, where the driving force of the syntactic process, the syntactic type, arises from the semantics of combinators. Chapter 3 presents argumenthood from the perspective of combinators. This is crucial for lexical capture of dependencies in the predicate-argument structure. It also suggests that the lexicon might be the source of undecidability if and when it is relevant. A more revealing aspect of combinators turns out to be what they deliver about discrete representability, rather than infinitude or decidability. Chapter 4 shows that the syntactic types of combinators cannot be arbitrary, due to having the same base for syntactic and semantic juxtaposition. Chapter 5 builds a substantive base on these formal foundations to present CCG as a linguistic theory. Chapters 6 through 8 discuss some variations in CCG theory: logical form (Chapter 6), possible constraints on all grammars (Chapter 7), and possible extensions of the invariants (Chapter 8). Chapter 9 evaluates some linguistic, philosophical, computational and cognitive aspects of CCG, all of which stem from bringing semantics into the explanation. Chapter 10 shows that CCG's computation must distinguish opaque and transparent processes, and that this leads to a syntactic simplification of its primitives to a single operation rather than two.

In conclusion (Chapter 11) a historical perspective is reiterated where adjacency as the sole hypothesis-forming device is singled out as CCG's most unique aspect, rather than variable elimination. This seems to be Schönfinkel's legacy.

Chapter 2

Order as constituent constructor

The semantic dependencies in a PADS must manifest themselves transparently in syntax for them to take part in constituencies and their interpretation, and for order (therefore adjacency) to remain as the only explanatory device for the syntax-semantics connection. This process will be called *syntacticization* throughout the book. The result is the embodiment of combinatory behavior in complex symbols called *syntactic types*.

1. Combinatory syntactic types

The notion of syntactic type has been imported to linguistic explanation, to the best of my knowledge, by Bar-Hillel, Gaifman and Shamir (1960), Montague (1970) and Gazdar (1981). Gazdar credits Harman (1963) for the first use of complex symbols in phrase-structure grammars, whereas Bar-Hillel et al's and Montague's use relates to Leśniewski's and Russell's types as functions.

Formally speaking, a type is a set of values. For example, we can think of the grammatical relation *subject* as a type, in English standing for the set of values {John, Mary, he, she, it...}. We can distinguish it from other types, say from the type *object*, which would be in English the set {John, Mary, him, her, it...}. We can also think of types for verbs, such as *tv* for transitives, which would be the set {hit, devour, read...}, and *iv* for intransitives, say {arrive, sleep, read...}. These sets can be countably infinite, which makes their finite representation by a type label even more significant.

Montague's deployment of a Russell-style type-theoretic language aims to give rise to meaningful expressions from a semantic type α (his ME_α), hence a simple label such as "subject" above would not do in his framework. His choice is to syntacticize a denumerable number of types α by building them into ME_α s. Such construal need not be variableless or order-induced.

As atomic labels in a phrase-structure grammar, types would bear no more significance than distributional notion of a category, such as N, V, A and P, for nouns, verbs, adjectives and prepositions, which are commonly employed in linguistics. This was the motivation for Harman (1963) to make complex

symbols first-class citizens of a phrase-structure grammar. In such complex symbols the notion of structure is assumed, rather than explained by adjacency.

What brings surface string generalizations of types from order-predicted semantics and syntax is the notion of a *combinatory syntactic type*, as employed in categorial grammars. For example, we can refine the type “subject” above as $S/(S\backslash NP)$ for English, which says that any value that takes a rightward VP as domain (because the label VP is typewise $S\backslash NP$), and yield a sentence as a result, belongs to the set of subjects. The “object” type would be different, for example $S\backslash(S/NP)$ for English. Although syntaxwise they differ, they arise from the same semantics, which is that of \mathbf{T} , because $\lambda P.Pa'$ is the semantics underlying this type, which means all functions P in which a' participates as an argument, which is a unary application of the combinator \mathbf{T} (1) to a .¹⁷

$$(1) \mathbf{T} \stackrel{def}{=} \lambda x \lambda y. yx$$

Syntacticization in this particular case refers to how the semantic dependencies engendered by \mathbf{T} directly imports to syntactic types such as $S/(S\backslash NP)$ and $S\backslash(S/NP)$ without further assumption. As shown in Chapter 4 the process is transparent, but it is not trivial, because syntactic dependencies carry different features than what is borne by semantic objects. For example, English subject-verb agreement spells the distinction $S/(S\backslash NP_{agr})$ for subjects and $S\backslash(S/NP)$ for nonsubjects where “agr” is a feature bundle for agreement. For Welsh, a strictly VSO language with subject-verb agreement, the distinction is between $S\backslash(S/NP_{agr})$ for subjects and $S\backslash(S/NP)$ for nonsubjects. Another lexical resource, the verb in this case, complements the picture by bearing the lexical type $S/NP/NP_{agr}$ for a Welsh transitive verb and S/NP_{agr} for an intransitive. These types are $(S\backslash NP_{agr})/NP$ and $S\backslash NP_{agr}$ for English.

The type $S/(S\backslash NP_{3s})$ for the word “Wittgenstein” syntactically denotes all functions that can be construed as an English speaker’s knowledge of all things “Wittgenstein” can grammatically do, in semantic terms, $\lambda P.Pwittgenstein'$, because it captures the following contrasts.

- (2) *Wittgenstein adores/*adore westerns.*
*Does/*do Wittgenstein adore westerns?*
*Milena writes more letters than Wittgenstein does/*do.*
*Wittgenstein I am sure takes/*take more notes than he publishes.*
*Wittgenstein you say is the one who adores/*adore westerns?*

*They adore/*adores Wittgenstein for that?*

*Wittgenstein I like/*likes, Russell I doubt.*

**?the film which might startle the critics and Wittgenstein would adore
the film which might startle the critics and which Wittgenstein would
adore*

We can also symbolize all things that can be predicated over “Wittgenstein”, in the syntactic type $S \setminus (S/NP)$ for an English speaker, which also has the semantics $\lambda P.P \text{ wittgenstein}'$. It takes another lexical resource to turn this into agreement. Because the English verb does not have $(S \setminus NP)/NP_{\text{agr}}$ for agreement, this possibility is avoided in English syntax. Therefore the category $S \setminus (S/NP)$ serves an entirely different syntactic function than $S/(S \setminus NP_{3s})$ of a subject participant. The knowledge of the word “Wittgenstein” is then construed as all possible categories that it can bear, in the form of syntactic type/predicate-argument structure pairs.

This construal of syntax-semantics correspondence can be compared with other type-dependent approaches. In Montague’s type system, where order does not step in to provide an interpretation, the type of a transitive verb is $((e, (e, t)), (e, t))$, which is model-ready for interpretation. In this sense, Montague’s Intensional Logic is dispensable as he pointed out himself (Montague 1970), in favor of a model-theoretic interpretation; see e.g. Dowty, Wall and Peters (1981) for discussion. In our case the type simply refines (or constrains) the correspondence of the syntactic type to its PADS. It is part of what computational linguists call a typed “glue language.”

2. Directionality in grammar: morphology, phonology or syntax?

The term *string type descriptor* for ‘:=’ in $he := S/(S \setminus NP_{3s})$ brings to mind whether we could entertain the possibility that some of these contiguous strings, namely words in the ordinary sense such as *adores* are derived combinatorially, or taken as axioms (lexical items) of a combinatory system. The first view is adopted here without elaboration. The second view would amount to taking ‘:=’ as the lexical type assignment operator. Equivalently we would be asking whether the word-internal compositional meaning assembly and constituency are mediated by the combinators as well, which is implicated by the view preferred here. I do not elaborate on it because the book covers no lexical dependency which refers to a part of another word.

The question brings forth the issue of morphology-phonology interaction during syntactic type-driven derivation. I will say nothing about these aspects in this book, because they need a book-length treatise of their own, which is upcoming work. Suffice it to say that we need to have a closer look at Separation Hypothesis in morphology (Beard 1987, 1995), that morphological and phonological types do form assembly, and syntactic-semantic types the meaning assembly. Modern morphological theories such as that of Lieber (1980), McCarthy (1981), Anderson (1992), Halle and Marantz (1993), Aronoff (1994), Beard (1995) and others need studying from a type-dependent perspective, to see if combinators are responsible for the meaning assembly in constructions involving parts of words and phrases.

Thus we will not be concerned whether the derivation of the following example from Arabic must compose the passive and the causative first, by **B** as shown, or whether we apply them one-at-a-time to the stem, which is also possible with the same type assumptions.

$$\begin{array}{c}
 (3) \quad \begin{array}{ccc}
 \begin{array}{c} -u- \\ \text{PASS} \end{array} & \begin{array}{c} -h- \\ \text{CAUS} \end{array} & \begin{array}{cc} \text{dahika} & \text{Ahmad Nadeem} \\ \text{laugh} & \begin{array}{c} \text{A} \\ \text{N} \end{array} \end{array} \\
 \hline
 \begin{array}{ccc}
 \frac{(S/NP/NP)/(S/NP/NP)}{\lambda P \lambda x \lambda y. \text{pass}'(Pyx)} & \frac{(S/NP/NP)/(S/NP)}{\lambda P \lambda x \lambda y. \text{cause}'(P(y))x} & \frac{S/NP}{\lambda x. \text{laugh}'x} \quad \frac{NP}{: a'} \quad \frac{NP}{: n'} \\
 \hline
 \begin{array}{c} -uh- := (S/NP/NP)/(S/NP) \\ \lambda P \lambda x \lambda y. \text{pass}'(\text{cause}'(P(x))y) \end{array} & \xrightarrow{\text{B}} & \\
 \hline
 \begin{array}{c} \text{duhhika} := S/NP/NP : \lambda x \lambda y. \text{pass}'(\text{cause}'(\text{laugh}'x)y) \end{array} & \xrightarrow{\quad} & \\
 \hline
 \begin{array}{c} \text{duhhika Ahmad} := S/NP : \lambda y. \text{pass}'(\text{cause}'(\text{laugh}'a')y) \end{array} & \xrightarrow{\quad} & \\
 \hline
 \begin{array}{c} \text{duhhika Ahmad Nadeem} := S : \text{pass}'(\text{cause}'(\text{laugh}'a')n') \\ \text{'Ahmad was made to laugh by Nadeem.'} \end{array} & \xrightarrow{\quad} &
 \end{array}
 \end{array}$$

Notice also the assumption that morphology and phonology somehow get it right that *-uh-* is a templatic infix to the verb stem. Crucially, the directionality of the slashes does not reflect morphology of Arabic. It is a syntactic constraint with a semantic motivation; in this case for example the passive looks for lexical verb categories.

We get grammatical derivations the same way in all languages by mediating them only through syntactic types, independent of their morphological or phonological typology, including for example the templatic morphology of Arabic, because syntactic dependencies relate to compositional semantics of words as they are embodied in syntactic types. The combinatory theory described in the book goes as far as claiming that the types above regulate the scope of e.g. the passive and the causative, because they are syntactic pro-

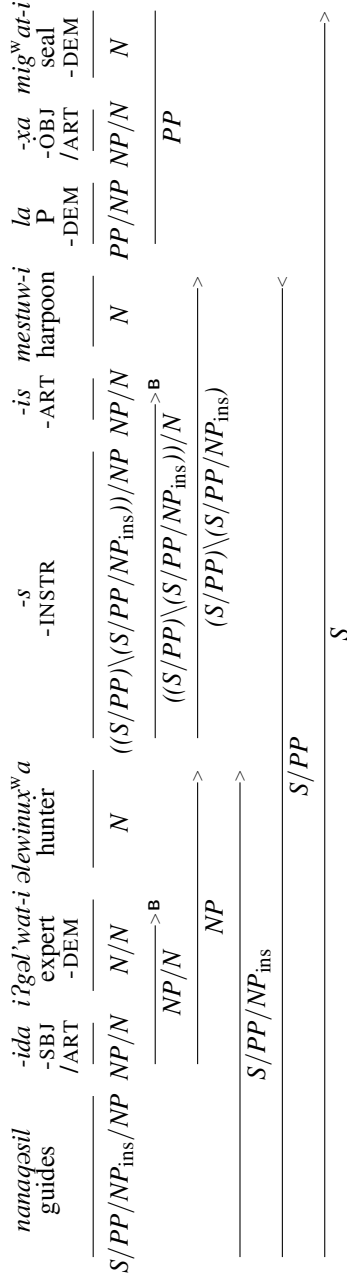
cesses. They regulate the behavior so that we get *pass'(cause'(laugh'd')n')* above, not *cause'(pass'(laugh'd')n')*, which is what we would also get if we let morphological types and phonology do the semantics, say by applying the passive $a \mapsto u$ to *dahika* first, and then the geminate causative to /h/. How these types arise from interfaces morphologically and phonologically is not covered in the book.

Further empirical support for dissociating syntactic directionality from morphological or phonological directionality comes from languages such as K^wak^wala where some nominal inflections fall on the preceding word, whatever its category. For example, in Figure 1, *-s* and *-is* are suffixes on *lewinux^wa* but they relate syntactically to *mestuw-i*. Similarly, *-ida* is a suffix on the preceding verb to which it bears no syntactic relation. The slashes in the figure reflect syntactic directionality rather than suffixation or morphological order.

In summary, the slash can only do syntactic work in a combinatory theory. If it takes on other duties such as morphological order (as it does in some versions of categorial grammar such as Hoeksema 1985), it cannot simultaneously undertake morphological work and afford not to immediately deliver semantics of some constituents. It would be forced to do that when composing the preceding word of an inflected nominal in K^wak^wala morphologically and phonologically because the semantics of the inflections would be unrelated to the morphological/phonological host. Positing phonologically vacuous types to remedy the problem would undermine the combinatory base of grammar because, in the process of syntacticization, only phonologically discernible elements can be given immediately deliverable semantics by combinators. Relaxing the directionality interpretation of a combinatory slash to allow syntactic, morphological or phonological order is not a degree of freedom in a combinatory grammar.

3. Trees and algorithms

The preceding discussion suggests that what we see in a combinatory derivation is a step-by-step syntactic and semantic assembly, not morphology or phonology. The style of the presentation wants explaining. Drawing the derivation in (3) as a tree reveals its strictly binary nature. This is shown in Figure 2. The same derivation could be drawn using the more familiar tree notation (Figure 3), but it would be misleading for three reasons.



'An expert hunter guides the seal with his harpoon.'

Figure 1. K^wak^wala's syntactic bracketing, adapted from Anderson (1992: 19).

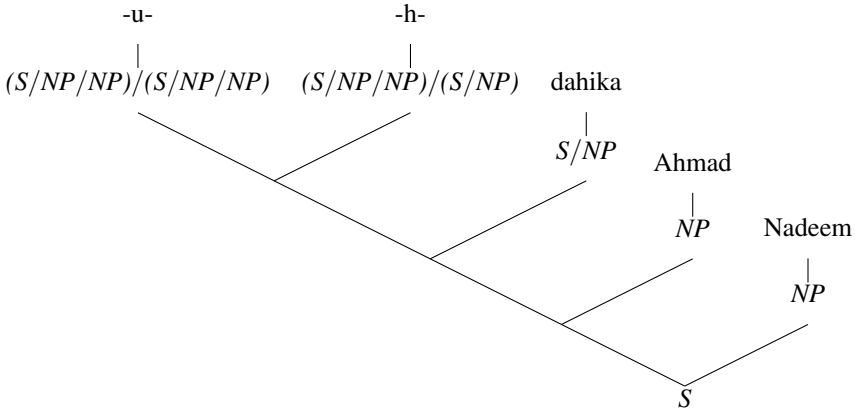


Figure 2. A CCG derivation as a tree.

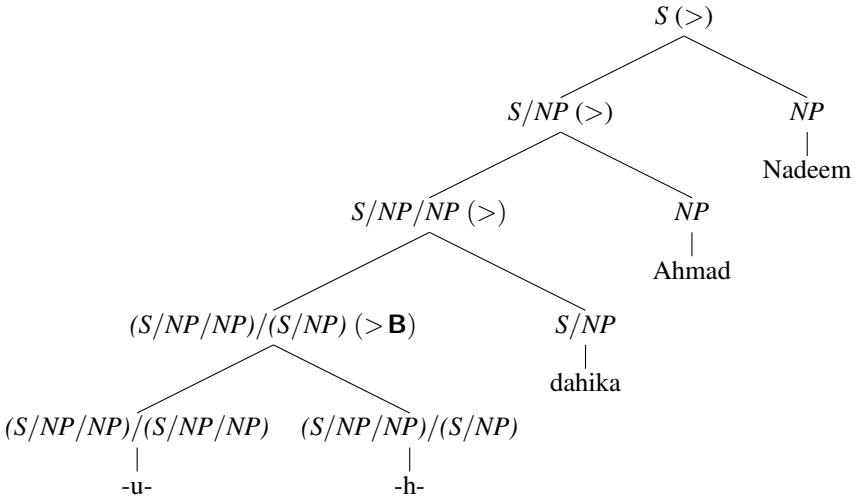


Figure 3. A CCG derivation as a phrase-marker tree.

First, structure-building in a combinatory derivation crucially depends on the linear sequence of types, which is explicit in a notation such as (3) but not in a phrase-structure tree.

Second, the combinatory process must start with the lexical assumptions. Otherwise there would be no way to achieve the immediate assembly of lexically projected semantics, whereas a tree can be built top-down, bottom-up or with a mixed strategy. In other words, a combinatory derivation is a con-

structive proof, an algorithm, of the structure-building, whereas a tree is its description.

Third, there are no intermediate records in a CCG derivation, which also breaks the ties with logical proofs. There is no sense in which any subtree would be available for reinterpretation, reuse, retraversal or reinspection.⁸ For example, after the derivation of *duhhika* above, we have the substrings *duhhika*, *Ahmed* and *Nadeem* as remaining work, without any rework or inspection. This is most explicit in line drawings, which can be viewed as walls built around the range of the derivation. I will use the standard linear notation throughout the book.

4. CCG's narrow claims in brief

Combinators as syntactic tools must encode and project dependencies just like they do when they operate on semantic objects. We must preserve this property throughout syntacticization so that we can claim the same origin (order) for structure and its interpretation. For example, a binary version of **B**, as in $\mathbf{B}fg = \lambda x.f(gx)$, suggests that f depends on g which depends on x , whatever x is when it is instantiated. No combinatory rule or dependency can change the dependence of f and g on x once we obtain $\lambda x.f(gx)$ by **B**. Parenthesis-free combinators such as **C** encode and project dependencies too. $\mathbf{C}fab = fba$, hence the order of the arguments matter to f in this example; it is a genuine dependency.

The syntactic process of combination might look similar in spirit to dependency grammars such as Tesnière (1959), Hudson (1984), Mel'čuk (1988). However, the narrower claim is that only the syntactic types bear on constituent structure, and they arise from semantics of order, therefore the process of syntacticization is crucial, and adjacency is all we need for it.

Having no degree of freedom from adjacency will force us to entertain narrow hypotheses about possible syntactic categories, therefore possible grammars, and about a basic inquiry of linguistics promoted in the preface:

(4) A Humean question for linguistics:

Why do we see limited dependency and constituency in natural language syntax?

Here is a brief preview of the limited constituency engendered by the syntacticized combinators. Although maximal left bracketing is allowed, not

Perhaps the best known work for variables in syntax is Ross's (1967) Coordinate Structure Constraint (CSC). His thesis was a bold attempt to constrain the syntactic variables. The motivation was to avoid overgeneration of the semantics of the constructions involving such kind of variables. Putting together the desire to constrain the semantic behavior, and employing syntactic variables for this task, we can conclude that these variables must range over structures, rather than strings or words.

Structure-dependence is the hallmark of transformationalism, both in the theory and in the data. Chomsky's transformations have changed over the years, but they have always maintained one property: structure preservation. According to this theoretical dictum transformations only apply to structured strings, represented as phrase-markers, to produce structured strings. In terms of data, assuming structure-dependence is the starting point for the nativist explanations of language acquisition (Crain and Pietroski 2001).

I will present structure-dependence and type-dependence in their own terms, and compare their claims. In the examples of structure-dependence below where the process of question formation pairwise relates a-examples to b-examples, the relevant relations are structural dominance and structural locality of labels.

- (6) a. *Kafka* [*liked Milena*]_{VP}.
 a'. *John* [*thinks that Kafka* [*liked Milena*]_{VP}]_{VP}.
 a''. [*The lady who I* [*think Kafka* [*likes*]_{VP}]_{VP}]_{NP} [*adored flowers*]_{VP}.
 b. *Did Kafka like Milena?*
 b'. *Does/*did John think that Kafka liked/*likes Milena?*
 b''. *Did/*do/*does the lady who I think Kafka likes/*like/*liked adore flowers?*

I use the notation []_T to represent the syntactic label *T* of the substring in brackets. For example, in (6a'), the inner VP is dominated by the outer VP. In (6a''), the outermost NP and the last VP are structurally sisters, hence local to each other. The stars in b'–b'' examples are meant to indicate that the meaning conveyed by an a-example cannot be questioned like the corresponding starred b.

If structural dominance were not critical, we would have the starred *do*'s in the b-examples as grammatical. If locality were not the determinant for the sisterhood of the subject, the starred *like* examples in b's would be fine too.

A simple inductive heuristic on the position of *do* or *like* ("for the choice of *do*, use a verb that appears later when the string is longer"), which might

work for (6a''), would not work for (7a). Similarly, a simple label match of *VP* by order would not work either (7b–c).

- (7) a. *The man who sleeps liked the lady who reads Kafka.*
 b. *Kafka* [[*while sleep*]_{VPing}]_{AdvP} *dreamed about Milena.*
 c. **Did Kafka while sleep dreamed about Milena?*

These examples are type-dependent as well as being structure-dependent. For example, we can think of yes-no questions as imposing the following constraints on *do*, where the syntactic labels are now combinatory syntactic types (constraints) rather than distributional categories.

- (8) a. [*Did*]_{S_{yn}/(S_{inf}\NP)/NP} *Kafka like Milena?*
 b. [*Does*]_{S_{yn}/(S_{inf}\NP)/NP_{3s}} *Kafka like Milena?*
 c. [*Do*]_{S_{yn}/(S_{inf}\NP)/NP_{-3s}} *you like Milena?*

With these assumptions, (9a) is ruled out by type-dependence without the help of structure-dependence. The inner *S_{inf}\NP* is not visible to the word *does*, and the string *think..Milena* cannot bear the syntactic type *S_{inf}\NP*.

- (9) a. **[Does]_{S_{yn}/(S_{inf}\NP)/NP_{3s}} Kafka* [*think* [*adore Milena*]_{S_{inf}\NP}]*?*
 b. *Do* [*you*]_{NP_{2s}} *think that* [*Kafka*]_{S/(S\NP_{3s})} *liked/likes/*like Milena?*
 c. *liked* := (S_{fin}\NP_{agr})/NP
 d. *likes* := (S_{fin}\NP_{3s})/NP
 e. *like* := (S_{fin}\NP_{-3s})/NP
 f. *like* := (S_{inf}\NP)/NP

Agreement is always encoded for subjects, as in *NP_{2s}* for *you* in (9b), also (*S_{yn}/(S_{inf}\NP)*)\((*S_{yn}/(S_{inf}\NP)*)/*NP_{2s}*), and for *Kafka*, as *S/(S\NP_{3s})*. This is enforced by the lexical differences in (9c–f).⁹ As in structure-dependent accounts, the category of embedded *likes* cannot project as the type of the clause headed by *think*. The critical type-dependent steps are shown below:

$$(10) \quad \begin{array}{ccccccc} \textit{Do} & & \textit{you} & & \textit{think that} & \textit{Kafka likes Milena?} & \\ \hline S_{yn}/(S_{inf}\backslash NP)/NP_{-3s} & & (S_{yn}/(S_{inf}\backslash NP))\backslash & & (S_{inf}\backslash NP)/S_{fin} & & S_{fin} \\ & & ((S_{yn}/(S_{inf}\backslash NP))/NP_{2s}) & & & & \\ \hline & & & & \xrightarrow{\hspace{10em}} & & \\ & & S_{yn}/(S_{inf}\backslash NP) & & & & S_{inf}\backslash NP \end{array}$$

Notice also that the choice of *liked* and *likes* in (9b) is not related transformationally as in (6a'/b'). They produce different semantics to begin with, which is a consequence of radical lexicalization. There are no deeper structures, with surface structures derived from them.

Structure-dependence and type-dependence begin to make different predictions when we observe that there might be (a) same structures which must bear different types, and (b) different structures which must bear the same type. In a type-dependent theory, different types mean differential behavior, and having the same type means manifesting the same syntactic behavior. The first kind is CCG's answer to CSC, without extraneous constraints, principles or variables. Let me briefly exemplify case (b) before we move to CSC. I will draw on Turkish data.

Common nouns and adjectives in Turkish are collectively called substantives because they show similar morphological characteristics when used as nouns, such as the same case, person and number marking. Their common semantics, that of being a property, which is syntactically *NP/NP*, is transparently imported to Turkish syntax in structures that widely differ in their internal structure but behave similarly in syntax.

We can for example form relative clauses which differ structurally in subject versus nonsubject extraction (11a–b), but both kinds can be headless as well, in which case they undergo the nominal paradigm in inflections as if they were noun stems (11c–d).

- (11) a. [*İstanbul'a gid-en*]_{NP/NP} *otobüs*
 Ist-DAT go-REL bus
 'The bus that goes to Istanbul' Turkish
- b. [*İstanbul'a git-tiğ-im*]_{NP/NP} *otobüs*
 Ist-DAT go-REL.1s bus
 'The bus with which I went to Istanbul'
- c. [[*İstanbul'a gid-en*]_{NP/NP}]_{NP}-*ler-i ben gör-me-di-m*.
 Ist-DAT go-REL-PLU-ACC I see-NEG-PAST-1s
 'I did not see the ones that go to Istanbul.'
- d. [[*İstanbul'a git-tik*]_{NP/NP}]_{NP}-*ler-im daha güzel-di*.
 Ist-DAT go-REL-PLU-POSS.1s more beautiful
 'The ones with which I went to Istanbul looked better.'

In these examples the headless variety cannot be thought of as cases where *biri* 'one' is deleted. For example, (11a) and (11c) are related and the readings are quantificational, but if we use *biri* or *şey* 'thing' in (11c), e.g. *İstanbul'a giden şeyleri ben görmedim* ('I did not see the ones that went to Istanbul'), it is nonquantificational. Therefore these are different structures. The examples have the additional property that, independent of the structural source, be

they a suffix, a lexically specified adjective (12), or a derived clause such as a headless relative clause, they can behave as anaphors if their type is a predicative NP.¹⁰ They have a unique semantic function syntactically.

- (12) [*Zengin*]_{NP/NP} *kriz-den etkile-n-me-di*.
 Rich crisis-ABL affect-PASS-NEG-PAST
 ‘The rich has not been affected by the crisis.’

In other words, Turkish seems to make no distinction in syntactic behavior of the types *NP/NP* and *NP* if the semantic origin of the *NP* is that of a property, independent of its internal structure. Compare the clausal structure of these examples with a nominal NP structure (13).

- (13) [*Her yeni otobüs-ün koltuğ-u*]_{NP}
 every new bus-GEN.3s seat-POSS.3s
 ‘every new bus’s seat’

The other case which differentiates type-dependence from structure-dependence is when similar structures show differential application in syntax, as in CSC.

Ross’s solution to CSC, that coordinands are islands of extraction with a single escape boat, which is to extract across the board (ATB) from each coordinand, and only for constituents with the same grammatical function in every coordinand, proved to require transderivational constraints for structure-dependent theories. No one has come up with an effective and nonarbitrary solution to such constraints which would keep the problem in the class of recursive languages describable by transformational grammars; see Peters and Ritchie (1973).

Through the syntacticization of combinators, the CSC becomes a type constraint without variables, kept well inside recursive languages; in fact it is nearly context-free. Here is the combinatory solution to the problem, as worked out mainly by Gazdar (1988) and Steedman (2000b). The type constraint is that the coordinands must be like-typed, enforced by the coordinator’s lexical category $(X \setminus X)/X$ in (14).¹¹

- (14) a. *The cat that [John admires]_{S/NP} and [Mary hates]_{S/NP}*
 b. **The cat that [John admires]_{S/NP} and [bites Mary]_{S \setminus NP}*
 c. **The man that [admires John]_{S \setminus NP} and [Mary detests]_{S/NP}*
 d. *The man [that admires John]_{N \setminus N} and [(that) Mary detests]_{N \setminus N}*
 Steedman (2011: 94)

- e. **The cat that [John admires]_{S/NP} and [Mary hates it]_S*
 f. **The cat that [John admires it]_S and [Mary hates]_{S/NP}*

The similarity of the argument to the structure-dependent explanation, that coordinands must be like-categories in the structural sense, is illusory; it is the computation of this constraint that makes structure-dependent theories Turing-complete, and type-dependent ones (in the combinatory sense) nearly context-free.

6. Constituency

Combinators as semantic objects cannot be the explanation why we see limited kinds of type dependencies in syntax. For example, we shall see that **S** can hardly be the explanation for the dependencies in *Mary wanted to love*, although they are certainly describable by **S**, because $\mathbf{S}fga = fa(ga)$, thus $\mathbf{S}(\mathbf{C}want')love'mary' = want'(love'mary')mary'$. But this combinator is precisely the syntactic explanation for the dependencies in *He is the man I will persuade every friend of to vote for*, and both reasons have to do with constituency as we shall later see.

Some dependencies are nonexistent semantically and syntactically, although they are describable by the combinators that operate in syntax. For example, there is no language in which the pseudo-English expression *John expects that Barry* could mean ‘John expects Barry to expect’. Its semantics would be $expect'john'(expect'barry')$. It is describable by **S**, **C** and **T**: $\mathbf{S}(\mathbf{C}\mathbf{C}john')(\mathbf{T}barry')expect','$ which is equivalent to the purported dependencies, $expect'(expect'barry')john'$. It will turn out to be a conspiracy of syntactic types of nominals and verbs, therefore not a theoretical impossibility but lexical improbability. The coconstraining behavior of syntactic types and semantics is a major concern of the book for this reason.

We need an agreed-upon definition of constituency to be able to judge the effects of semantic dependencies on syntactic grouping.

I will follow an empirical notion of constituency, which is assumed to be the basis of competence:

- (15) Any surface string with compositional semantics that can be put together phonologically by a native speaker is a constituent.

As an empirical requirement, it says that whenever we observe an intonational grouping which is acceptable by native speakers, we must worry about

its compositional meaning, and about how to deliver that meaning. As a theoretical requirement, it says no more than that every syntactic combination that mediates the phonology-semantics connection must have a semantic interpretation, otherwise we would just have a mixture of words rather than constituents, a point which Chomsky (1975: 206–211) was the first to point out back in 1955.

This definition and its theoretical and empirical aspects seem to be shared by transformationalism and other frameworks as well. Consider for example Chomsky's criteria for phrase-markers, which embody constituency in his theory ever since its inception.

- (16) 1. The rule for conjunction Chomsky (1975: 210)
 2. Intrusion of parenthetical expressions
 3. Ability to enter transformations
 4. Certain intonational features.

Chomsky goes on to argue in the next page that the first and the second criteria are actually theoretical, and can be subsumed by the third, but the fourth criterion is not. Therefore we are forced to have at least one theoretical and one empirical criterion for constituency, which is followed here as well.

In a theory where structures are classified by subtyping, such as HPSG, constituency is directly built into the theory. Phrasal types are distinguished from lexical types by subtyping, with the further division of phrasal types as headed structures and others. Only the subtypes of the type *phrase* carry a feature called DAUGHTERS, subtyped as constituent structure (their *construc*), Pollard and Sag (1994: 31). Because all types have a semantic feature as well, it is incumbent on an HPSG grammar to show a head for the headed constituent structures, and no head for others, which establishes a good empirical test for constituency.

The concept is manifest in multistructural theories of grammar such as LFG, as “*order-free composition*, requiring that the grammatical relations that the [grammatical] mapping derives from an arbitrary segment of a sentence be directly included in the grammatical relations that the mapping derives from the entire sentence, independently of operations on prior or subsequent segments,” Bresnan and Kaplan (1982a: xlv). The nature of the mapping is the theoretical claim, and the inclusion of grammatical relations is the empirical test. LFG culminates the resolution of these multiple constraints on an independent level, called c(onstituent)-structure, with each level having its own well-formedness conditions. Their point extends to assigning a syntactic

mapping to the following fragments, just like complete sentences, precisely because the theory can show how their grammatical relations can be included in the set of interpretations of the larger segment of which they are a part:

- (17) a. *There seemed to ...* Bresnan and Kaplan (1982a: xlv)
 b. *...not told that...*
 c. *...too difficult to attempt to...*
 d. *...struck him as crazy...*
 e. *What did he...*

In summary, there seems to be a consensus that constituency must have a theoretical foothold and an empirical testing ground, without which it seems hard to formulate a grammar. Using a variableless, monostratal, order-instigated syntax for this task, which is presented here, and its way of handling constituency, naturally brings to mind comparisons to syntax with variables, most notably with transformationalism, which as its name suggests needs variables.

Consider the two different analyses of *the man who Mary loved*, shown below. (18) is an analysis based on Steedman's Combinatory Categorical Grammar (Ades and Steedman 1982, Steedman 2000b).

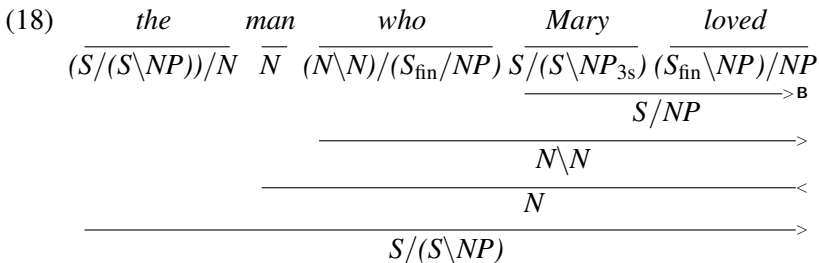


Figure 4 uses a recent version of transformationalism, the Minimalist Program, which started with Chomsky (1993, 1995).

The analysis with variables, Figure 4, uses six primitives: move, merge, agree, check, lexical insertion, and argument structure. The last one ensures that we get a merge of *loved* and the syntactic variable *-wh*, rather than just *loved*, as in *Mary loved deeply*. Its scope is controlled by the governor *+wh*. Lexical insertion injects parts of words into the tree, and ensures for example that there is one copy of *Mary*.

A structure-dependent but order-inspired theory of structure-building, that of Phillips (2003), appeals to order as its main thrust of the construction operation, and likewise uses several copies of words (first created then deleted

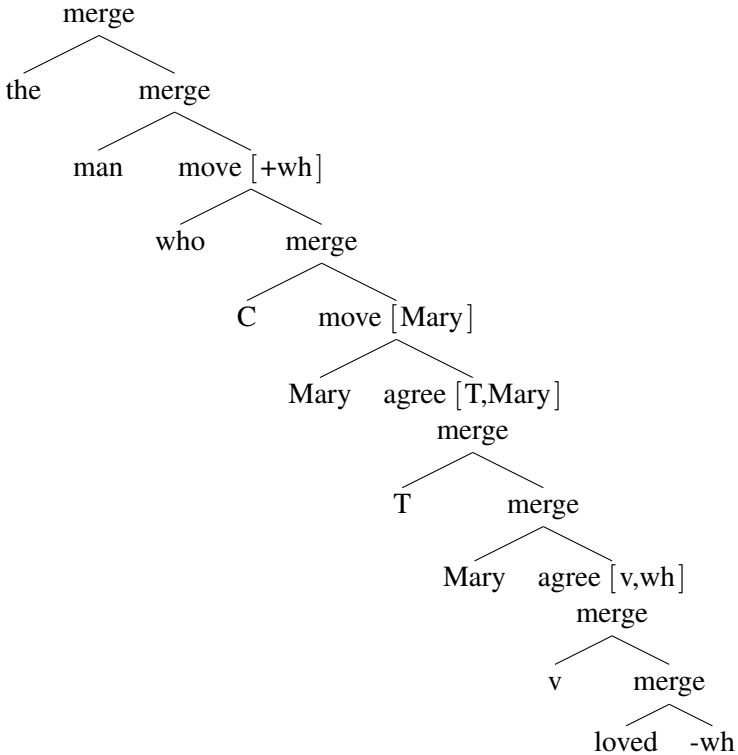


Figure 4. Minimalist Program’s primitives.

under identity), plus the operations move, merge and the economy conditions on structures. It is not monotonically dependent on the syntactic types of the words in a sequence.

The purpose of the book is to show that (18) uses only one primitive: Schönfinkel’s juxtaposition. Every syntactic combination is local and adjacent. It is meaning-bearing, and phonologically realized. For example, **B**’s syntacticization arises from its dependency structure, written after a colon, which I use for the time being to talk informally about semantics.

$$(19) X/Y: f \ Y/Z: g \ Z: a \rightarrow X: f(ga)$$

We could not conceive a **B** semantics if the syntactic types were one of the following in (20). Either adjacency (20a–c) or dependency (20d–e) are violated in these configurations.

- (20) a. $*X/Y \ Y \ Y/Z \rightarrow X$
 b. $*Y/Z \ X/Y \ Z \rightarrow X$
 c. $*X/Y \ Y/Z \ P/Q \ Z \rightarrow X \ P/Q$
 d. $*X/Y \ Y/Z \ Z \rightarrow Y$
 e. $*X/Y \ Y/W \ Z \rightarrow X$

Syntactic types adhere to dependency by virtue of adjacency as well. From the derivational configuration $A \ B \Rightarrow C$, shown on the left below, which means the syntactic types A and B given in this order leads to the syntactic type C , we can also obtain the same result by assuming $A=C/B$ and $B=C \setminus A$:

$$(21) \quad \frac{A \quad B}{C} \qquad \frac{A=C/B \quad B}{C} \qquad \frac{A \quad C \setminus A = B}{C}$$

For example, the English transitive construction ‘NP V NP’ spells a combinatory type for the verb as follows: ‘NP V NP’ $\Rightarrow S$, hence ‘V NP’ $\rightarrow S \setminus NP$.¹² Therefore ‘V’ $\rightarrow (S \setminus NP) / NP$. A phonological string α with the morphological type ‘V’ is known to syntax only by its syntactic type $(S \setminus NP) / NP$.¹³ We write this as:

$$(22) \quad \alpha := (S \setminus NP) / NP$$

Other translations are possible, for example $(S / NP) \setminus NP$ for ‘V’, but this category is easily eliminated by the litmus test of syntax, surface constituency: (23a) is grammatical, therefore its surface constituents must be derivable with the verbal category assumptions.

- (23) a. *Obelix (chases relentlessly) and (eats ferociously) the wild boars of the Armorican forest.*
- b. $\frac{\frac{\text{eats}}{(S/NP) \setminus NP_{3s}} \quad \frac{\text{ferociously}}{(S \setminus NP) \setminus (S \setminus NP)}}{?} \quad \frac{\frac{\text{eats}}{(S/NP) \setminus NP_{3s}} \quad \frac{\text{ferociously}}{(S/NP) \setminus (S/NP)}}{<B}$
- c. $\frac{\frac{\text{John fights ferociously.}}{NP_{3s} \ S \setminus NP_{3s} \ (S/NP) \setminus (S/NP)}}{?}$

A category such as $(S / NP) \setminus NP$ for transitives would not be consistent or complete, because we must assume a consistent and complete category for the adverbial as well. Compare (23b) and (23c). The adverbial assumption in the first alternative of (23b) would be unworkable with the verbal assumption, as shown. The second alternative on the right is workable, but it would be insufficient for the constituency in (23c). The remaining potential culprit is the verbal assumption in (23), which must be revised. The categories $(S \setminus NP) / NP$

and $(S \setminus NP) \setminus (S \setminus NP)$, respectively for the verb and the adverb, are consistent and complete with respect to the observations of constituency above.

The argument structure arises from adjacency too. There is a systematic relation between a syntactic type such as $(S \setminus NP) / NP$ of *love* and its dependency representation $\lambda x \lambda y.love'xy$, which we can eta-normalize without variables to $love'_{(e,(e,t))}$. Similarly, the $S \setminus NP$ of the intransitive *love* and its semantics $\lambda x.love'x$, which we can normalize to $love'_{(e,t)}$, are codeterminant.

A purported argument structure in the category $(S \setminus NP) / NP : \lambda x.love'x$ is universally disallowed, only because its eta-normalized version, $love'_{(e,t)}$ which is variableless, could not give us a complete interpretation of the verb. There are two syntactic slashes, therefore two syntactic arguments, hence we must expect two lambdas (perhaps more, as in properties, but at least two, because of the syntactic type). Although we can associate the variable x with the $' / NP$, rightly or wrongly, there would be no semantic counterpart of $' \setminus NP$ above, which is to say that we have no way of capturing its meaning because we would have no way of knowing what syntactic objects (words, phrases) it is argument of by virtue of adjacency. This cannot be the competent knowledge of the word *love*, whether it is $love'_{(e,(e,t))}$ or $love'_{(e,t)}$.¹⁴

Both syntax and semantics work by juxtaposition. Indeed, semantics becomes immediately available at every step of the derivation because of having the same primitive. I redraw the derivation of (18) below to show the lock-step assembly of semantics driven entirely by syntactic types.

$$\begin{array}{l}
 (24) \quad \begin{array}{cccccc}
 the & man & who & Mary & loved & \\
 \hline
 (S / (S \setminus NP)) / N & N & (N \setminus N) / (S / NP) & S / (S \setminus NP)_{3s} & (S_{fin} \setminus NP) / NP & \\
 : \lambda P \lambda Q.(the'x)and'(Px)(Qx) & : man' & \lambda P \lambda Q \lambda x.and'(Px)(Qx) & : \lambda P.Pmary' & : \lambda x \lambda y.loved'xy & \\
 & & & & S / NP : \lambda y.loved'ymary' & \xrightarrow{B} \\
 & & & & N \setminus N : \lambda Q \lambda x.and'(loved'xmary')(Qx) & > \\
 & & & & N : \lambda x.and'(loved'xmary')(man'x) & < \\
 \hline
 S / (S \setminus NP) : \lambda Q.(the'x)and'(and'(loved'xmary')(man'x))(Qx) & & & & & \xrightarrow{}
 \end{array}
 \end{array}$$

Notice that the process of lexical insertion into phrase-structural intermediate records (trees) is replaced by a process of bringing the self-contained type assignments of the meaning-bearing elements to the surface string. They cannot be copied, checked or governed, and there can be no late or early insertion. Such devices need structure-builders over and above order, as Phillips's (2003) work demonstrated.

It is a prediction of a lexical insertionless theory such as CCG that morphological and phonological assembly interact with grammatical computation in limited ways, to affect the syntactic types only at the interfaces. This

issue is related for example to Chomsky's "derivation by phase" (Chomsky 2001). CCG's conjecture is that a phase has a very limited window of opportunity, namely one meaning-bearing item in the string, regulated by its lexical syntactic type (Steedman 2005b). This makes "phase" synonymous with 'a lexical item that can be spotted in a string, one with a syntactic type and a predicate-argument dependency structure'.

In this sense, the theory of CCG is not derivationalist in its account of constituency and interpretation, because no condition can be predicated over derivations if there aren't any intermediate records to predicate over. Representationalism, which is a term commonly used in transformational studies to show the contrast in their way of management of intermediate results, such as Brody (1995), Epstein et al. (1998), is not helpful to characterize CCG either. It can best be characterized as a type-dependent (rather than structure-dependent), radically lexicalist approach to syntax which relies on adjacency as the only structure building primitive, and only in places where structure truly manifests itself: surface constituency and predicate-argument structure.¹⁵

CCG's principle of adjacency is not an argument of theoretical simplicity or Occam's razor. Chomsky's point on the topic of theory choice is well-taken:

"Thus it is misleading to say that a better theory is one with a more limited conceptual structure, and that we prefer the minimal conceptual elaboration, the least theoretical apparatus. [...] If enrichment of theoretical apparatus and elaboration of conceptual structure will restrict the class of possible grammars and the class of sets of derivations generated by admissible grammars, then it will be a step forward (assuming it to be consistent with the requirement of descriptive adequacy)." Chomsky (1972: 68)

The program of CCG is bringing semantics into the explanation in a completely syntactic type-driven grammar and its computation. If semantics can reduce the possible lexical categories hence possible grammars, without further auxiliary assumptions, then its role in the explanation might be considered a complication in the theory for a good reason. (It would be a complication because the semantic representation is now part of the knowledge we can collectively call a *category*, together with the syntactic type.) If a significant reduction can be shown, then a narrower theory is to be preferred. However, doing this the CCG way shifts the goals of linguistic theorizing from narrowing down the admissible phrase markers to understanding the limited nature

of dependency and constituency despite the apparent flexibility in order and structure. Hence the question is more complex than presented so far.

The insistence on adjacency distinguishes CCG from theories which are otherwise similar in spirit in adopting lexicalism and the abandonment of transformations. For example, HPSG had in the past posited empty strings in the lexicon for topicalization and relativization (Pollard and Sag 1987), then moved towards the elimination of traces (Pollard and Sag 1994). LFG has this option too; cf. Kaplan and Bresnan (1995), Kaplan and Zaenen (1995). Type-logical grammar can assign types to empty strings and retract such assumptions under certain conditions, or stay away from this practice as it sees fit regarding semantics, e.g. Carpenter (1997).¹⁶ CCG has no such degree of freedom.

The notion of *possible grammars* can be equated with *possible combinatory categories* when we insist on adjacency and radical lexicalization because only lexical items can bear categories and the categories contain no variables. Combinatory constituency is the litmus test for such categories. A related cousin of juxtaposition called “wrap” does not provide a combinatory base, as we shall see in §5.1.

Chapter 3

The lexicon, argumenthood and combinators

Let us now see how combinators can capture function-argument configurations as a consequence of juxtaposition, and without variables. This will give us a variableless lexicon. Then we move on to variableless syntax. First, some history of the variable.

Peirce's (1870) elimination of variables predates Frege's decisive work on clarifying the notion of variable, and Peirce was apparently unaware of Frege's work. Frege's (1891) variableless technique was to represent for example $x^2 + x$ as $()^2 + ()$. The notation, as he prophesied, did "not meet with any acceptance" (Frege 1904:p.114). His currying in Frege (1893) is almost identical to what we have now, due to its adoption by Church (1936) for lambda calculus. Frege's program aims to distinguish intensions such as $()^2 + ()$ from extensions (values) such as $\lambda x.x^2 + x$.

The two notations put together did not lend themselves to purely adjacency-driven models of semantic object manipulation. Schönfinkel had to appeal to Łukasiewicz-style prefix notation to facilitate variableless combination by adjacency.

However, he did not use Łukasiewicz's (1929) prefix operator—which Quine 1967 symbolized as o , to represent $x(yz)$ as $oxoyz$. He used the parenthesized notation instead. Thus Quine (1967) is right to criticize Behmann for adding the end material to the 1924 paper about the elimination of parentheses, which Schönfinkel apparently had not intended as his agenda.

It is sometimes useful to make a clarification about the whole practice of variable elimination. As Curry pointed out frequently (Curry 1929, 1963, Curry and Feys 1958), combinatory logic concerns itself with the elimination of variables from elementary theorems, but leaves open the question of their utility in epitheorems. Thus a foundation is set in which we can safely assume that bound variables, if used, are used only for expository or efficiency purposes (because of Church-Turing thesis and the equivalence of lambda calculi and combinators—see Barendregt 1984). Steedman (1988, 1996a) suggests that bounded constructions (passives, reflexives etc.) are one area in which a variable-friendly logical form in an otherwise variableless combinatory syntax might have evolutionarily arisen out of pressures for efficient processing.

1. Adjacency and arity

We can now move toward a variableless lexicon in Curry's sense of eliminating them from fundamental theorems. An n -argument predicate f can be uniquely represented as f^n if we wished. However, the arity declaration of an object is an intrinsically combinatory property of it, therefore the f^n notation would not do to establish the lexicon-syntax communication by order alone. Curry and Feys's (1958) definition of power for combinatory objects reveals the right combinatory source. We can define the arity of f as a consequence of juxtaposition. It marks the arity of f as a combinatory prefix.

$$(1) A(f, n) \stackrel{\text{def}}{=} \begin{cases} f & \text{for } n = 0 \\ \mathbf{B}^n \mathbf{I} f & \text{for } n > 0 \end{cases} \quad (\text{Schönfinkel-Curry arity})$$

Some manifestations of combinatory arity are exemplified below.

$$(2) \begin{aligned} fabcde\dots & \qquad \qquad \qquad (f^0) \\ \mathbf{B}^1 \mathbf{I} f abcde\dots & = \mathbf{I}(fa)bcde\dots = (fa)bcde\dots & (f^1) \\ \mathbf{B}^2 \mathbf{I} f abcde\dots & = \mathbf{BB} \mathbf{I} f abcde\dots = \mathbf{I}(fab)cde\dots = (fab)cde\dots & (f^2) \end{aligned}$$

Because of \mathbf{I} , every abstraction is necessarily a function if there are arguments. This is implicit in Schönfinkel's notation §1(1).¹⁷

The notation translates to syntactic argument-taking directly; the power of \mathbf{B} in (1) is the number of slashes of f in its syntactic category. For $\mathbf{B}^3 \mathbf{I} f$, we get for example $A/B/C/D$ for f where A is the result type of f , but not $A/(B/C)/D$, because the second slash in the latter category would be for the argument of B , not A . Similarly, if f is a zero-argument function (a constant), then $\mathbf{B} \mathbf{I} f$ or $\mathbf{I} f$ would not faithfully reflect that it is not necessarily a functor; it can be say A rather than A/B , hence the first clause of (1).

The reason for going through the trouble of variable-free argument specification is to show that argument taking is just another manifestation of semantic dependency, and to show that the adjacency formulation of dependency finds a natural niche for it in syntax without being orthogonal to, or an auxiliary assumption of, phrase structure. All of the combinators' behavior is describable solely by the adjacency of functions and arguments. \mathbf{S} , \mathbf{B} and \mathbf{I} etc. can take their arguments only if they are adjacent. The results are predictable directly from their adjacency. The dotted material in for example $\mathbf{B} f ab \dots d \dots$ is "unreachable" to \mathbf{B} , therefore uninterpretable by this \mathbf{B} . The object d cannot be an argument of this \mathbf{B} , by the virtue of its nonadjacency. The objects f , a and b must be the arguments of \mathbf{B} because of their adjacency.

The combinators **B**, **S**, **I** etc. are assumed to contain no vacuous abstraction in their definition, i.e. all and only the arguments are specified. Hence there is no version of **B** which is able to reach out and take the d above via vacuous abstraction, say $\lambda x_1 \cdots \lambda x_n . x_1(x_2 x_4)$ for some $n > 3$. This is not a theoretical necessity because by definition any object is a combinator if it has no free variables, including the ones with spurious abstractions. It is in this sense that we take them as ‘building blocks’ as Schönfinkel had called them; all other combinatory definitions are illative.¹⁸

Now a single grammatical base, adjacency, explains all behaviors of argument-taking objects because we know from Curry and Feys (1958) that combinators have the same power as the lambda calculus. (This is somewhat tolerable in computing, but it presents problems to a linguistic theory. I say more on this in the closing words of this chapter.)

The effect of unification of argument-specification and combinatory behavior under adjacency might be quite revealing for the radical lexicalization of natural grammars. Most importantly, we get full interpretability of words and phrases, which is what argument specification is all about, for free. This result arises from *supercombination*, along with finite typeability in the lexicon.

2. Words, supercombinators and subcombinators

For the purpose of understanding linguistic argument-taking by combinators, the relation between combinatory terms and lambda terms requires a closer look. For example, $f(\lambda x . g(hx))$ indicates that x is not an argument of f but of h . If f is the head functor of a word w , then this lambda term suggests that x is not an argument of w but of some other word which f takes in its domain. An example of such dependency is the bracketed substring in [*what you can*] and *what you must not count on*.

Assuming $\lambda Q . ?yQy$ for the semantics of *what* for simplicity, following Hoyt and Baldridge (2008), Groenendijk and Stokhof (1997), the substring encodes the dependency in (3a), but not (3b).

- (3) a. *what you can* := $\lambda P . ?y can'(Pyyou')$
 b. $*\lambda P \lambda x . ?y can'(Pxyou')y$

In other words, *what you can* is a one-argument function, not two. The variable y is a nonsyntactic argument of P . (P in this case corresponds to *count*

on, whose predicate-argument structure $\lambda x_1 \lambda x_2. \text{can}' x_1 x_2$ is opaque to *what* and *what you can*.) The difference arises from the nature of combinators and supercombinators.

All the combinators we have seen so far are supercombinators, with the exception of **O** and **Y**, to be defined below. Supercombinators can group their argument abstractions—lambdas—to the left to leave a lambdaless body. This seems to be a clear identification of a predicate-argument structure in a category, where the lambdas can be seen as the glue language for syntactic arguments. We will have a closer look at **Y** later because it is crucial for the debate on syntactic versus semantic recursion.

The opacity in *what you can* arises from **O**. In combinatory parlance, this combinator is not a supercombinator : $\mathbf{O} f g h \stackrel{\text{def}}{=} f(\lambda x. g(hx))$. The semantics of *what you can* requires this combinator: $\mathbf{O} \text{what}'(\text{you}' \text{can}')$, where $\text{what}' = \lambda Q. ?y Q y$. A preview of CCG's syntactic type-driven way of handling this dependency is given below along with its semantic assembly (4). It makes use of the syntacticized **O** rule in (5). I will justify the syntactic types of (5) in the next chapter.

$$(4) \quad \frac{\frac{\frac{\overline{S/(S/NP)} \quad \overline{S/(S \setminus NP)} \quad \overline{(S \setminus NP)/(S \setminus NP)}}{\lambda Q. ?y Q y} : \lambda f. f \text{you}' \quad \lambda P \lambda x. \text{can}'(Px)}{S/(S \setminus NP)} : \lambda P. \text{can}'(P \text{you}')}{S/((S \setminus NP)/NP)} : \lambda P. ?y \text{can}'(P \text{you}')}{\text{O}}$$

$$(5) \quad \text{a. } X/(Y/Z): f \quad Y/W: g \quad W/Z: h \rightarrow X: f(\lambda x. g(hx)) \quad (\mathbf{O})$$

$$\text{b. For some } X/Y: h \quad (\mathbf{Y})$$

$$X/Y: h \quad \leftrightarrow \quad \mathcal{F}_0 = X/(X/Y): \mathbf{Y} h$$

$$(X/Y)/\mathcal{F}_{n-1} \quad \leftrightarrow \quad \mathcal{F}_n \quad \text{for } n > 0$$

The significance of supercombinators for our purposes is the following.

- (i) Inner lambdas cannot be the arguments of f in (5a), hence the ternary nature of **O**, although there is a lambda left on the right-hand side of its definition. **Y** is considered unary for the same reason.

Therefore argumenthood is not a simple count of lambdas. It is a structural property because it requires the knowledge of inclusion asymme-

tries. This is one of the reasons why we need the notion of predicate-argument structure in addition to dependency, leading to PADS.

- (ii) Words whose semantics require combinators which are not supercombinators can be called *subcombinators*. Although they may look odd as words, such as *what you can*, with **O** semantics as shown above, they can in principle be lexicalized. For example, the Turkish equivalent of *that I defended* is indeed one word, *savunduğum*, which also has **O** semantics as we shall later see. They necessarily absorb an argument of their arguments because of an inner lambda abstraction, as in x of $\mathbf{O}fgh \stackrel{\text{def}}{=} f(\lambda x.g(hx))$.

Not all subcombinators are finitely typeable. **O** has finitely many types, but **Y** does not (5); notice the recurrence relation in **Y**. Finite typeability seems to be a prerequisite for compositional semantics of words because it translates to lexical representability.

- (iii) The words with subcombinator semantics must be distinguished from function words whose arguments may be opaque in a different way. For example, in languages where unbounded relativization is headed by a relative pronoun, such as *that* in English, we have the semantics $\lambda P\lambda Q\lambda x.\text{and}'(Px)(Qx)$ for the relative marker. Here opacity arises from the fact that x substitutes for a property in Q and a participant in P , cf. *the dog that the cat chased* versus **Fido that the cat chased*. There are no inner syntactic lambdas in $\lambda P\lambda Q\lambda x.\text{and}'(Px)(Qx)$; it is indeed a supercombinator.
- (iv) Words whose semantics demand a lexical use of combinators such as **Y** would be very odd. If there were such words, their combinatory behavior could not be read off entirely from their argument types because the syntactic contexts in which they can occur cannot be known fully by the native speaker; note the recursive variable \mathcal{F} in (5) above. It is tantamount to saying that knowledge of these words cannot be complete. We can conjecture that no such word exists in natural languages. Therefore,
- (v) The only lexicalizable dependencies that are manifest in natural language are the finitely typeable ones. They are describable by supercombinators and subcombinators. This is a necessary but not sufficient condition. We shall see examples such as the combinator **K**. It is a

finitely typeable supercombinator which is very unlikely to be operating in syntax or in the lexicon.

3. Infinitude and learnability-in-principle

Clearly, a subset of combinators ought to be considered as potential combinatory apparatus for a linguistic theory. The dependencies manifested by **Y** and **K** have not been attested in natural languages, and **C** might wreak havoc in grammar but perhaps not in the lexicon. CCG has a specific answer to this problem, which I summarize in Chapter 5.

In a way linguistics faces the same amount of problems meeting the combinators when physics faced against Roger Penrose's claim that classical physics is Turing-computable: none.¹⁹ It did not make the Turing machine a rival theory of classical physics, because it cannot predict anything unless physicists engage substantive constraints in their theory. Similarly, combinators cannot be a theory of language just because they happen to be the models of adjacency par excellence. This is where the linguistic theorizing begins for combinators.

Three issues arise for any linguistic theory aspiring for formal adequacy and substantive restrictiveness: infinity, decidability and representability of natural language. The combinatory perspective suggests that, although all three issues are crucial, representability is the most decisive among the three, and it is not some informal notion of representability, but Turing representability. The reasons are as follows.

The argument for the infinitude of human languages first appealed to Cartesian creativity and von Humboldtian romanticism, respectively: (a) there is a universal repertoire of thoughts with infinite ways to express them, and (b) individual languages materialize as the special manifestations of a universal human language. Chomsky's (1966) integration of these two lines of thought as the cornerstones of his generative grammar "of infinite use of finite means" carried the finiteness debate into the realm of formal methods.

Generative grammar attempted to enumerate possible grammars, but the earlier attempts were overshots. Putnam (1961) criticized the basic innovation of generative grammar, transformations, as being able to generate nonrecursive languages, and maintaining that human languages are recursive. Putnam's claim had been criticized to be too performance-oriented, but Peters and Ritchie (1973) argued from the perspective of competence grammars,

and could not find a nonarbitrary way of delimiting possible transformational grammars to guarantee a constrained formalism.

Chomsky's theorizing shifted away from formal aspects by the early 60s, and the debate on the undecidability of his formalism faded.²⁰ He claimed that recursion is the basic trait of human language, for example Chomsky (2000), Hauser, Chomsky and Fitch (2002). The notion of recursion is most formally dealt with in mathematics and computing science, and the results I summarize in §4.1 and §9.2 suggest that what Chomsky seems to have in mind is everybody's assumption, that semantic recursion, i.e. recursion by value, is real for all humans. Syntactic recursion, however, i.e. recursion by a name or a label, is not necessary for this, and the lack of a **Y**-like behavior in any natural language can be taken as the living proof of this result. **Y** is the paradoxical combinator of Curry, and without it or its behavioral equivalent such as Turing's **U**, syntactic recursion is not possible, as we shall see in §4.1.

Pullum and Scholz (2009) argue that giving up on recursion is not a mental block to creativity. After all, 10^{230} might be the number of possible sentences in human languages, and it does require a theory to sift through the search space to identify say English, even though the search space is finite.

I am of course not suggesting that we take the easiest way out to satisfy Gold's (1967) finding about learnability, by assuming that languages are learnable because they are finite. In his "text" model where the acquirer faces the same conditions as the child, only finite languages can be learned. In the other model, called the "informant", any grammar up to and including that of primitive recursive languages can be learned.²¹ The model requires a decider to answer whether a string is in the language or not. Gold himself acknowledges that it requires feedback about negative instances "by being corrected in a way we do not recognize" Gold (1967: 453).

The computationalist scenarios I outline in §9.5 suggest that there is probably more indirect evidence than what is assumed by the complex innate knowledge proposals. For example there is the possibility of the child being wrong about what an utterance means, but being very explicit about the syntax-semantics connection hypothesis, for example thinking that *veggies* means *dog'* when the word is uttered when there is a dog around, or that *veggies* is an act like eating, with a syntactic type such as *S/NP* rather than *NP* as the adult might have intended. The indirect evidence here might be the next state of affairs where there are veggies but no dogs around, or no potential for being forced to eat them, such as being pointed in a grocery display while sitting in a stroller. Infinitude seems to be a secondary concern in this task.

But learning “something more than the data” in the Humean sense does prove critical; see §9.5 for discussion, where something more is claimed to be the syntactic type.

Without too much of a worry about finitude, we can readjust the goals of linguistic theory to understand why we do not see some kinds of dependencies and constituencies in any language, whether they are finite or not. Free operation in syntax, and the codetermination of syntax and semantics in the form of a category, seem to suffice for this line of research.

Now let us consider decidability. A weak argument arises from formal aspects, such as transformationalism not being able to deliver grammars that always decide. We do not know whether this is the reason why Chomsky (1965) entertains the possibility of natural languages being potentially undecidable.²² One formalization of minimalist grammars, that of Stabler (1997, 1999), suggests that Chomsky’s recent grammars stay well within recursive languages.

A stronger argument is from languages rather than grammars. A naive version of the argument might proceed as follows: human languages are decidable because every speaker can decide whether any expression is a sentence in her language. Differences of opinion would not count because the speakers would have to make up their minds in the first place to be able to agree or disagree.

What makes them decidable is a meta-theoretical question, but it would not lead to a theory of language if it fails to engage substantive constraints in a linguistic theory. Levelt (1974) suggests that one such constraint is the learnability-in-principle, which amounts to saying that acquirable grammars are the primitive recursive ones. This is one of the running themes of this book, and it requires a closer look at substantive constraints on grammars, which we will narrow down to a theory of possible lexical categories. We know that a concocted language in which every sentence has an even number of words is decidable, yet there is no such language and we can be certain that there will never be. So what is unnatural about this language? Clearly, no amount of formalization can give us the desired answer, because the very word *natural* requires that we situate the formal apparatus in some complex system with interactions, i.e. a system with substantive constraints.

We can also entertain the possibility that human languages may be Turing-undecidable but Putnam-Gold decidable. Putnam-Gold (1965) machines are Turing machines that can change their minds—if you pardon the expression—as the computation develops. Thus, for a known Turing-

undecidable problem, a Putnam-Gold machine can output a “no” before computation begins, and then output a “yes” or another “no” depending on which state it halts. If it never does, we still have an answer.²³ It does not follow that any undecidable problem can be modeled that way. Take for example the question:

(6) *What is the next real number after π ?*

The argument of decidability *for* language must remind us that language is not posing that kind of a question to us, even though the question may be very relevant to the semantics out there, where meaning cannot be determined by language.

This brings us to the final issue, that of finite versus transfinite representability. There is another argument of undecidability that we should take into account in this regard, that of Hintikka (1977). He uses the semantic criterion of synonymy, of interchangeability of *any*-expressions with *every*-expressions in English, which he shows to be not even recursively-enumerable. Bresnan and Kaplan (1982a: xlv) comment that “If Hintikka’s argument is correct, then semantics must diverge from syntax in a fundamental way, as he observes.” Remember also Quine’s (1951) warning that the notion of synonymy brings with it other problematic concepts such as analyticity.

We could also speculate whether the problem as stated by Hintikka is Turing-representable in the first place. Bolinger (1968: 234) offers another linguistic perspective in this regard, which suggests that we might start with questioning Hintikka’s experiment and its implications for the nature of semantic representation: “Practically speaking, there is no such thing as an identical synonym. The language demands its money’s worth from every word it permits to survive.”

Where does synonymy stop, if it exists? (Note that in the work cited above and in the follow-up Hintikka 1980, the test requires *any*-substituted sentence to be grammatical, *and* contrast in meaning.) In this continuous space of similarity, we can also include problems that are not Turing-representable. For example, if and at what level can we say that a cat is sitting on the mat? At the folk science level or ordinary language, with some nominal understanding of sitting, we can test this hypothesis, but at the quantum level? Some of the quantas of the cat *might* be communicating with the mat to a level we might consider touching, but surely not all of them. How that experiment differs from synonymy experiment is not clear. (See Higginbotham 1982 for another

kind of objection, that taking logical equivalence as a sufficient condition for sameness of meaning is problematic. Quine's demonstration of circularity of analyticity and synonymy presents a conundrum for semantic criteria as well.)

Thus any criterion of decidability ought to be syntactic and combinatorial, otherwise we are in a domain much like the real numbers, and we can forget about a combinatorial base for language.

The moral of the thought experiment is that it pays to keep the problem combinatorial by sticking to a syntactic criterion of (un)decidability. We know some realizable classes of formal machines to see what kind of computational resource management we need to capture the kinds of dependencies we see in natural languages. We have no such hope as yet for transfinite representations which are implicit in (6). The notion of representability is dubious in that domain.

In this context, limited noncontext-freeness of human languages formally argued for by Shieber (1985) and Joshi (1985) provides a research agenda in which the limited nature of the automaton itself is the explanation for the limited kinds of dependencies, rather than extra assumptions or stipulations. This also cuts down severely the degrees of freedom in theorizing because limited computational resources can be called in for help in a hypothesis. In this way of thinking going in the syntactic route all the way to undecidability would not change the underlying syntactic machinery, it would just mean that the source of undecidability might be the lexicon, such as a word with **Y** or **WWW** semantics.

Thus, Turing representability in the abstract is the key to be able to even talk about the syntactic manifestation of semantic dependencies. Words with syntactic dependencies *are* the observables on which we can theorize about semantics. Decidability and finiteness are secondary issues.

That of course does not entail that the biological substrate of the limited automaton is the answer to our combinatorial problems. There is a very likely possibility that the (human) brain is not a sequential computer like the Turing machine. For all we know, the underlying cognitive mechanism *for* language may not be language-specific at all. And this is where the linguistic theorizing stops, in case the warnings of Sandra (1998) about what linguists can and cannot say about human language processing mechanisms are not clear enough, with our current level of understanding. Remember the debate in the 1990s about the psychological reality of traces and empty categories. For every experiment which proved the reality of such elements (see Zurif 1995,

Gibson and Hickok 1993), there was a counter-experiment which proved their nonexistence (e.g. Pickering and Barry 1991, Pickering 1993).

This takes us back to variables in theorizing. Traces and empty categories are syntactic variables in need of binding or government. Why eliminate them when they are so convenient to our understanding of argument-taking? Computing scientists face the same predicament for different reasons. The computing story is quite revealing, but I leave it to programming language theorists to tell that story.²⁴

In this book I will stick to the linguistic story. One of the most striking empirical observations of the 20th century linguistics is that parsing is a reflex. (Try turning it off if you are a skeptic, and imagine someone saying the ineffable as you try to shut yourself down.)²⁵ It is tempting to say that we could import computing's success with variableless interpretation to account for the reflex-like behavior of knowledge of language in action (the key word here is *like*, because the metaphor seems to fail in predictable ways in for example aphasia and autism).

A less speculative answer is that the kind of combinatorics that is revealed before us in the form of syntacticized combinators sets up a base on which substantive theories can be built to predict possible linguistic categories, therefore possible languages. The adjacency base of semantics directly translates to adjacency syntax when we eliminate variables from fundamental theorems.

The interesting turn of variableless theorizing with combinators is that not only do they suggest a formal source for the combinatory possibilities in languages, they make the combinations—constituents—directly and immediately interpretable if the ingredients happen to have semantics. That is the bread and butter of a competence grammar, and we get a modeling tool in which syntax and semantics coconstrain possible lexical categories to provide a substantive base.

And everything does have semantics, including the so-called dummies (for example the *it* in *It seems to rain*), the accusative case and function words such as *that*, *to* etc., once we readjust our semantic radar.²⁶ The purpose of the book is to show an attempt of that model building process in detail.

Chapter 4

Syntacticizing the combinators

The combinators were originally intended to deal with functions. For them to do syntactic-semantic—i.e. grammatical—work, we need their faithful translation into syntactic objects so that the semantic dependencies they symbolize are directly imported into syntactic dependencies. This is what I mean by syntacticizing the combinators.²⁷

The reader might object that what I call “functions” are syntactic objects, because lambda calculus and variableless combinators seem to manipulate them by syntactic rules.

They may be called syntactic objects of a domain theory, i.e. a name for collection of objects, but they would not be the syntactic objects of a linguistic theory. Consider the same problem (levels of abstraction) for the theory of lambda calculus. It has a direct denotational semantics for any lambda expression, for example x denotes all values of x in an environment e , $\lambda x.M$ denotes all values denoted by M when the free occurrences of x in M gets some value, say a . Lambda terms are its syntactic objects, and sets-as-denotations are its semantic objects. (See Barendregt 1984, Stoy 1981 for a full treatment of denotation and its relation to the syntax of lambda calculus.)

We face the same levels of abstraction problem in combinatory linguistics. Although a compositional meaning of the phrase *love hurts* could be given as **B***hurt'**love'* if we wished, this must arise from words as syntactic objects, since we cannot communicate combinatory thoughts as combinatory thoughts. (If you are not convinced, try conveying the meaning of *love hurts* without words, in a medium in which you must also be able to convey the meanings of: *I believe love hurts. Mary claims I believe love hurts. The man in the corner claims Mary thinks I believe love hurts.* etc.)

This brings us to the ontology of objects in a linguistic theory. CCG's handling of dependency is different from that of dependency grammars, where it is taken as an asymmetric relation among words (syntactic objects) in a string. In CCG, the dependency relation is defined over semantic objects, but since the observables are syntactic objects, the relation must be mediated by syntactic types. This might be considered a complication in the theory in Chomsky's sense noted earlier, but it is for a good reason: it can give us predictions about surface constituents and their immediate interpretability.

We first syntacticize application. The slash ‘/’ is the syntactic counterpart of function application, which is made explicit in Schönfinkel-Curry arity §3(1), where the power of \mathbf{B} translates to the number of slashes for arguments. We write $\mathbf{B}^1\mathbf{I}f$ as $A/B : f$. The syntactic type of f states that it is syntactically a function from B to A .

We can now syntacticize the semantic dependency manifested by juxtaposition fa :

$$(1) X/Y : f \quad Y : a \rightarrow X : fa \quad (\text{application})$$

‘ \rightarrow ’ is the syntactic counterpart of the reduction rule, viz. beta-conversion. There is no restriction that Y be slashed or slashless. This follows from the semantics of application, which is (fa) but not necessarily $f(\mathbf{I}a)$.

We write (2) syntactically to mean that the syntactic objects ω_1 and ω_2 , with categories $A/B : f$ and $B : a$, capture the semantic dependency fa in their syntactic types.

$$(2) \frac{\frac{\omega_1}{A/B} \quad \frac{\omega_2}{B}}{A} \text{-app}$$

Argument-taking objects such as f above are curried functions. Thus every such f takes one argument at a time. Its syntactic type cannot be slashless because, if it could, we could write application as (3) as well (a ‘*’ in a rule decoration indicates ill-formedness).

$$(3) X : f \quad Y : a \rightarrow X : fa \quad (*\text{application})$$

There is nothing in the ingredients of the rule (3) that says f is the function and a is the argument, yet the result requires it. The rule is not compositional as it stands. The X/Y type for $\mathbf{B}^1\mathbf{I}f$ forces a function interpretation on the syntactic side as well, hence the rule (1).

The syntactic type of $\mathbf{B}^n\mathbf{I}f$ has n slashes as in $X/_1 \cdots /_n Y$. The last slash is the one relevant to (1), because the left-associativity of juxtaposition naturally translates to the left-associativity of the slash.²⁸ $X/_1 \cdots /_n Y$ is same as $(X/_1 \cdots)/_n Y$.

The application rule cannot be (4a) either because the semantic dependency is fa , not af . (4b) fails to capture the dependency of f ’s argument type and a . Z cannot be an arbitrary argument type; it must be Y .

$$(4) \begin{array}{ll} \text{a. } Y : a \quad X/Y : f \rightarrow X : fa & (*\text{application}) \\ \text{b. } X/Y : f \quad Z : a \rightarrow X : fa & (*\text{application}) \end{array}$$

Thus the only syntacticized rule of application that translates the semantic dependencies to syntactic dependencies without further assumption is (1). We can write (1) as (5) because of this result, and fully syntacticize it.

$$(5) X/Y \quad Y \rightarrow X \quad \text{(application)}$$

Table 1 lists all the combinators which Curry and Feys (1958) considered more or less basic. Smullyan (1985) retold the story of combinators as talking birds, presumably anticipating their natural fit with language.²⁹ The names in the third column are Smullyan's birds. We shall syntacticize them—and more—one by one.

Table 1. Basic combinators

I	I $x = x$	Identity bird
Y	Y $x = y = xy$ for some y depending on x	Sage bird
K	K $xy = x$	Kestrel
T	T $xy = yx$	Thrush
W	W $fx = fxx$	Warbler
B	B $xyz = x(yz)$	Bluebird
C	C $xyz = xzy$	Cardinal
S	S $xyz = xz(yz)$	Starling
Φ	Φ $xyzw = x(yw)(zw)$	
Ψ	Ψ $xyzw = x(yz)(yw)$	
J	J $xyzw = xy(xwz)$	Jay

1. Unary combinators

The first unary combinator is **I**. We can syntacticize it as (6). Unary rules are simple correspondences without combination, which we write with a double arrow.

$$(6) X/Y : a \leftrightarrow X/Y : \mathbf{I}a \quad \text{(I)}$$

At first sight **I** might look superfluous because it adds nothing to the inventory of semantic objects or syntactic types. It does crucial work on the lexical side when we want to ensure that an argument of an object is an argument-

taking object itself. On the syntactic type, such constraints translate to requiring a slashed category. For example, f can be typed $A/(B/C)/(D/E)$ if both arguments are unsaturated functions (remember that currying will take care of the arity of B and D). Thus the following purported syntacticization of \mathbf{I} does not import the semantic property that whatever a is, $\mathbf{I}a$ is necessarily a syntactic and semantic function.

$$(7) X : a \leftrightarrow X : \mathbf{I}a \quad (*\mathbf{I})$$

The other unary combinator, \mathbf{Y} , which was discovered by Curry, is the epitome of recursion, and rightfully established him as the father of functional programming by the 1970s.³⁰ For example, \mathbf{YK} deletes infinitely many objects. Curry and Feys (1958) called it *the paradoxical combinator* because it captures Russell's paradox nicely. It is better known as the *fixpoint* combinator, which allows recursive programs to be written without variables or names. Recall that \mathbf{Y} behaves the following way: $\mathbf{Y}h = h(\mathbf{Y}h)$.

Not surprisingly, \mathbf{Y} 's syntacticization fares no better than infinite regress in semantics, and leads to an infinite schema:

$$(8) \text{ For some } X/Y : h \quad (\mathbf{Y})$$

$$\begin{aligned} X/Y : h &\leftrightarrow \mathcal{F}_0 = X/(X/Y) : \mathbf{Y}h \\ (X/Y)/\mathcal{F}_{n-1} &\leftrightarrow \mathcal{F}_n \quad \text{for } n > 0 \end{aligned}$$

What makes the syntacticized \mathbf{Y} syntactically recursive is the recurrence relation \mathcal{F}_i , not having the same result as its argument, as in $X/(X/Y)$. This observation will be crucial in the following chapters.

We can see the syntactically recursive behavior of \mathbf{Y} in (9), for a hypothetical word ω .

$$(9) \quad \begin{array}{c} \omega \\ \hline X/Y : h \\ \hline \mathcal{F}_0 = X/(X/Y) : \mathbf{Y}h \\ \hline \mathcal{F}_1 = (X/Y)/\mathcal{F}_0 = (X/Y)/(X/(X/Y)) : h(\mathbf{Y}h) \\ \hline \mathcal{F}_2 = (X/Y)/\mathcal{F}_1 = (X/Y)/((X/Y)/(X/(X/Y))) : h(h(\mathbf{Y}h)) \\ \hline \mathcal{F}_3 = (X/Y)/\mathcal{F}_2 = (X/Y)/((X/Y)/((X/Y)/(X/(X/Y)))) : h(h(h(\mathbf{Y}h))) \end{array}$$

The property that saves the infinite expansion from unwarranted undecidability is what computing scientists call *lazy evaluation*, which is to avoid evaluating an argument in normal-order until it is demanded by its function.

It is a consequence of the Church-Rosser (1936) theorems. No one has identified a word in any language that requires the second derivation line above. Thus, although **Y** can be kept under control by lazy evaluation, no such dependency seems manifest in languages.

2. Binary combinators

Let us now consider Schönfinkel's binary combinators **T** and **K**. **T** can be syntacticized as (10). By **T**'s semantics, viz. $\mathbf{T}ab = ba$, we know that b is the function and a is the argument.

$$(10) Y: a \ X/Y: b \rightarrow X: \mathbf{T}ab \quad (\mathbf{T})$$

We cannot have (11) as the syntactic reflexes of **T**. The overall syntactic type is that of b , viz. X , which is not guaranteed in (11a). (11b) fails to capture **T** semantics because $a \neq \mathbf{T}a$. **T** wants the function after the argument.

$$(11) \text{ a. } Y: a \ X/Y: b \rightarrow Z: \mathbf{T}ab \quad (*\mathbf{T})$$

$$\text{ b. } X/Y: b \ Y: a \rightarrow X/(X/Y): \mathbf{T}a \ X/Y: b \quad (*\mathbf{T})$$

T's syntacticization is completed once the semantic dependencies are directly reflected in the syntactic types. We can rewrite (10) without semantic objects from now on:

$$(12) Y \ X/Y \rightarrow X \quad ({}_2\mathbf{T}=\mathbf{T})$$

We can carry over the X/Y of (12) to the right to fully syntacticize the unary version of **T**:

$$(13) Y \leftrightarrow X/(X/Y) \quad ({}_1\mathbf{T})$$

What allows us to do this is the asymmetry of juxtaposition inherent in Schönfinkel's interpretation, that the sequence ab is not the same as the sequence ba , thus $Y \ X/Y$ is not the same as $X/Y \ Y$. Therefore, carrying over the Y in (12) to the right, for example as $X/Y: b \rightarrow X \setminus Y: b$, would be wrong, whereas $X/Y: b \rightarrow X \setminus Y: \lambda x.bx$ is fine.³¹ (The backslash attempts to keep the relative order of X/Y and Y .) The equivalence of the first case would imply $ab = ba$ necessarily. The relation must be mediated, and $\mathbf{T}ab = ba$ is a way of doing that.

We can see the effect of importing the mediation to syntactic types in the following examples: (14a–b) embody **T** semantics, whereas (14c) does not.

$$(14) \text{ a. } \frac{\frac{\omega_1}{Y} \quad \frac{\omega_2}{X/Y}}{X}^{\mathbf{T}} \quad \text{b. } \frac{\frac{\omega_1}{Y} \quad \frac{\omega_2}{X/Y}}{X/(X/Y)}^{\mathbf{T}} \quad \text{c. } \frac{\frac{\omega_1}{Y} \quad \frac{\omega_2}{X \setminus Y}}{X}^{\mathbf{T}}$$

$$\frac{\quad}{X}^{\text{app}}$$

However, there is a systematic relation between the forced \mathbf{T} semantics of the kind in (14a–b), and optional \mathbf{T} semantics in (14c). This is shown in (15). From this perspective, \mathbf{T} can be seen as the application of an argument as a function in one direction, to a function which looks for an argument of that kind in the other direction.

$$(15) \frac{\frac{\frac{\omega_1}{Y: a} \quad \frac{\omega_2}{X \setminus Y: f}}{X/(X \setminus Y): \mathbf{T}a}^{\mathbf{T}}}{X: \mathbf{T}af = fa}^{\text{app}}$$

It is called *type raising* for this reason, which necessarily involves applicative configurations:³²

$$(16) \begin{array}{l} X/Y \quad Y \rightarrow X \quad \quad Y \leftrightarrow X \setminus (X/Y) \quad (\text{type raising}) \\ Y \quad X \setminus Y \rightarrow X \quad \quad Y \leftrightarrow X / (X \setminus Y) \end{array}$$

The process is order-preserving, and relaxing this property results in permutation closure (Moortgat 1988a). The optionality of \mathbf{T} proves to be a necessary degree of freedom in the account of flexible constituency, as we shall see in Chapter 5.

\mathbf{K} 's syntacticization is straightforward because it does not follow from a semantic dependency between its arguments:

$$(17) X: a \quad Y: b \rightarrow X: \mathbf{K}ab = a \quad (\mathbf{K})$$

\mathbf{K} 's power of deletion is unmatched by any of the combinators in Table 1, therefore it is not interdefinable by these combinators or juxtaposition. Its unary version serves to show its formidable powers, by freely deleting the syntactic dependencies of any Y :

$$(18) X: a \leftrightarrow X/Y: \mathbf{K}a = \lambda b.a \quad ({}_1\mathbf{K})$$

The last binary combinator in Table 1 is \mathbf{W} . With semantics $\mathbf{W}fa = faa$, it can behave incessantly like \mathbf{Y} in certain circumstances such as \mathbf{WWW} . By definition, f requires two arguments. We can syntacticize it as follows:

$$(19) (X/Y)/Y: f \ Y: a \rightarrow X: \mathbf{W}fa \quad (\mathbf{W})$$

It would be wrong to syntacticize it as below. (20a) would turn a one-argument f into a two-argument f . (20b) would not be compositional: there is no indication that the second argument— Y —is reduced to the first argument Z , hence the semantic dependency of \mathbf{W} is not wholly reflected in the syntactic types.

$$(20) \text{ a. } X/Y: f \ Y: a \rightarrow X/Y/Y: f \ Y: a \ Y: a \quad (*\mathbf{W})$$

$$\text{ b. } (X/Y)/Z: f \ Z: a \rightarrow X: \mathbf{W}fa \quad (*\mathbf{W})$$

Carrying over the Y from the left-hand side of (19) to the right-hand side, and writing the remainder as \mathbf{W} capture the semantics of \mathbf{W} (21a), which we can fully syntacticize as in (21b).

$$(21) \text{ a. } (X/Y)/Y: f \rightarrow X/Y: \mathbf{W}f = \lambda a.faa$$

$$\text{ b. } (X/Y)/Y \leftrightarrow X/Y \quad ({}_i\mathbf{W})$$

The reader will note the lavish use of resources by \mathbf{W} , and wasteful \mathbf{K} . When applied to semantic objects, say $\mathbf{K}fa$ to waste a , or $\mathbf{W}fa$ to bring another a out of a hat, this may look tolerable. But when the objects in question are syntactic objects, namely words, resource insensitivity takes on a whole new meaning. We shall see in subsequent chapters that resource sensitivity does not necessarily follow from adjacency (witness \mathbf{K}), therefore exclusion of \mathbf{W} or \mathbf{K} from syntax must be scrutinized, rather than assumed because of their resource insensitivity.

3. Ternary combinators

We now turn to combinators with three arguments. The one with the simplest semantics is \mathbf{B} the compositor, which embodies the composition of two functions: $\mathbf{B}fga = f(ga)$. We can syntacticize it as follows:

$$(22) X/Y: f \ Y/Z: g \ Z: a \rightarrow X: f(ga) \quad (\mathbf{B})$$

Notice that, by definition, f and g must both be argument-taking objects because they occupy the functor position. This can be made more explicit by writing their semantics as $\mathbf{B}(\mathbf{I}f)(\mathbf{I}g)a = f(ga)$, of which (22) is a direct translation with slashes.

For us to get the same **B**-dependencies as syntactic dependencies, the following must be eliminated; f must depend on a because it depends on g which depends on a :

$$(23) X/W: f \quad Y/Z: g \quad Z: a \rightarrow X: f(ga) \quad (*\mathbf{B})$$

B's syntactic manifestation as (22) is redundant because of the primitive (juxtaposition). This effect can be seen below where the task of **B** is done by two applications of the primitive on the right. This notion of redundancy will be crucial in Chapter 5 where we choose the free combinators for syntax.

$$(24) \begin{array}{ccc} \omega_1 & \omega_2 & \omega_3 \\ \hline X/Y & Y/Z & Z \\ \hline X & & \end{array} \mathbf{B} \qquad \begin{array}{ccc} \omega_1 & \omega_2 & \omega_3 \\ \hline X/Y & Y/Z & Z \\ \hline Y & & \\ \hline X & & \end{array} \text{-app}$$

The following manifestation of **B**, in which the right edge component of (22) is carried over to the right-hand side, is nonredundant. We can take (25b) to be the syntacticization of the semantic dependencies in (25a).

$$(25) \begin{array}{l} \text{a. } X/Y: f \quad Y/Z: g \rightarrow X/Z: \mathbf{B}fg = \lambda x.f(gx) \\ \text{b. } X/Y \quad Y/Z \rightarrow X/Z \end{array} \quad (2\mathbf{B})$$

The following translations of (22) are wrong, because the redundancy due to ternary application is purportedly eliminated by carrying over the middle argument to the right. In $\mathbf{B}fga$, **B**'s semantics is lost if g is after a .

$$(26) \begin{array}{l} \text{a. } X/Y: f \quad Z: a \rightarrow X/(Y/Z): f(g) \quad Y/Z: g \quad (*\mathbf{B}) \\ \text{b. } X/Y: f \quad Z: a \rightarrow X/Z: \lambda x.fx \quad Y: ga \quad (*\mathbf{B}) \\ \text{c. } X/Y: f \quad Z: a \rightarrow X/(Y/Z): \lambda g.f(ga) \quad (*\mathbf{B}) \end{array}$$

The adjacency constraint on f, g, a in $\mathbf{B}fga$ is violated in the following example: Z is unreachable to X/Y and Y/Z to be interpreted by them. It would be a nonadjacency semantics for **B**.

$$(27) X/Y: f \quad Y/Z: g \quad W: h \quad Z: a \rightarrow X: f(ga) \quad W: h \quad (*\mathbf{B})$$

Thus the only nonredundant syntacticization of **B** which preserves the semantic dependencies is (25b). We can produce the unary version from (25b) as well, which will help us to simplify the syntacticization of other combinators. The right periphery of the left-hand side in (25b) can be carried over to the right-hand side as long as we maintain the right order of arguments, as in (28). This is what Curry and Feys (1958) called $(\mathbf{B})_1$.

$$(28) X/Y \leftrightarrow (X/Z)/(Y/Z) \quad ({}_1\mathbf{B})$$

Next we consider **C**, the elementary permutator, with semantics $\mathbf{C}fba = fab$. This combinator swaps the order of arguments for an argument-taking object f . Although it does not introduce parentheses on the right, **C** is a dependency encoder, unlike **K**, which is another parenthesis-free combinator. The function f depends on the arguments a and b , and their change of order is significant to f . It can be syntacticized as follows:

$$(29) (X/Y)/Z: f \quad Y: b \quad Z: a \rightarrow X: fab \quad (\mathbf{C})$$

The first argument of f must be of the same type as the second argument in linear order, hence the purported syntacticization in (30) cannot preserve **C**-dependencies engendered by the types of arguments and their adjacency.

$$(30) (X/Z)/Y: f \quad Y: b \quad Z: a \rightarrow X: fab \quad (*\mathbf{C})$$

$$(31) \frac{\frac{\omega_1}{(X/Y)/Z} \quad \frac{\omega_2}{Y} \quad \frac{\omega_3}{Z}}{X}^{\mathbf{C}} \quad \frac{\frac{\omega_1}{(X/Y)/Z} \quad \frac{\omega_2}{Y} \quad \frac{\omega_3}{Z}}{\frac{Z/(Z/Y)}{(X/Y)/(Z/Y)}^{\mathbf{B}}}}{X/Z}^{\mathbf{B}} \quad \frac{\quad}{X}^{\text{app}}$$

C's ternary manifestation is behaviorally equal to unary **T**, binary **B**, unary **B** and application (31). Similarly, its binary version (32a) is equivalent to the behavior of the same combinators (32b). Unary **C** is defined in (33).

$$(32) \text{ a. } (X/Y)/Z \quad Y \rightarrow X/Z \quad ({}_2\mathbf{C})$$

$$\text{ b. } \frac{\frac{\omega_1}{(X/Y)/Z} \quad \frac{\omega_2}{Y}}{\frac{Z/(Z/Y)}{(X/Y)/(Z/Y)}^{\mathbf{B}}}}{X/Z}^{\mathbf{B}}$$

$$(33) (X/Y)/Z \leftrightarrow (X/Z)/Y \quad ({}_1\mathbf{C})$$

The syntacticization of **S** the substitutor follows a similar line. Unlike **B**, the combinator **S** assumes a two-argument f in $\mathbf{S}fga = fa(ga)$. (Schönfinkel had called it *fusion*, which makes the dependency of both functions on the remaining argument very explicit.)

We can faithfully reflect the arity and adjacency of the arguments of **S** in the following syntacticization.

$$(34) (X/Y)/Z: f \quad Y/Z: g \quad Z: a \rightarrow X: fa(ga) \quad (\mathbf{S})$$

We cannot conceive the following configuration as **S** because it amounts to having $Sfaga = fa(ga)$ for some **S**. This is different than $\mathbf{S}fga = fa(ga)$.

$$(35) X/Y/Z: f \quad Z: a \quad Y/Z: g \quad Z: a \rightarrow X: fa(ga) \quad (*\mathbf{S})$$

The following purported syntacticizations of **S** are wrong because they do not embody **S** semantics. The first one violates the dependency of both f and g on a . The second one violates the adjacency of f and g in $\mathbf{S}fga$.

$$(36) \text{ a. } (X/Y)/Z: f \quad Y/W: g \quad W: a \rightarrow X: fa(ga) \quad (*\mathbf{S})$$

$$\text{ b. } (X/Y)/Z: f \quad Z: a \quad Y/Z: g \rightarrow X: fa(ga) \quad (*\mathbf{S})$$

Ternary **S**'s work can be done by the syntacticized combinators **W**, **B** and **C**. Curry and Feys (1958) note the equivalence $\mathbf{S} = \mathbf{B}(\mathbf{B}(\mathbf{B}\mathbf{W})\mathbf{C})(\mathbf{B}\mathbf{B})$. Smullyan (1985) gives a simpler formula, $\mathbf{S} = \mathbf{B}(\mathbf{B}\mathbf{W})(\mathbf{B}\mathbf{B}\mathbf{C})$. These combinators are explicit in the right column of (37).

$$(37) \begin{array}{ccc} \omega_1 & \omega_2 & \omega_3 \\ \hline (X/Y)/Z: f \quad Y/Z: g \quad Z: a & & (X/Y)/Z: f \quad Y/Z: g \quad Z: a \\ \hline X: fa(ga) & & (X/Z)/Y: \mathbf{C}f \\ & & \hline & & X/Z/Z: \mathbf{B}(\mathbf{C}f)g \\ & & \hline & & X/Z: \mathbf{W}(\mathbf{B}(\mathbf{C}f)g) \\ & & \hline & & X: \mathbf{W}(\mathbf{B}(\mathbf{C}f)g)a = fa(ga)^{\text{app}} \end{array}$$

The binary and unary versions of **S**, derived from (34), are as follows:

$$(38) (X/Y)/Z \quad Y/Z \rightarrow X/Z \quad ({}_2\mathbf{S})$$

$$(X/Y)/Z \leftrightarrow (X/Z)/(Y/Z) \quad ({}_1\mathbf{S})$$

4. Quaternary combinators

Let us now consider the combinators with four arguments. The first one is Φ , with the semantics $\Phi fgha = f(ga)(ha)$. It can be syntacticized as follows. Note that f is a two-argument function, and g and h must be functions.

$$(39) X/W/Y: f \quad Y/Z: g \quad W/Z: h \quad Z: a \rightarrow X: f(ga)(ha) \quad (\Phi)$$

It would be wrong to syntacticize it as (40), because the semantics of Φ would not be ensured on the right-hand side: $\lambda y.gy =_{\alpha} \lambda x.gx$, but locally substituting the behaviorally equivalent $\lambda y.gy$ loses the semantics of Φ , viz. the same a for g and h .

$$(40) \quad (X/Z)/(W/Z)/(Y/Z): f \quad Y: g \quad W: h \quad Z: a \rightarrow \\ X: \lambda x.f(\lambda x.gx)(\lambda x.hx) [x/a] \quad (*\Phi)$$

Thus the semantics of Φ is intrinsically related to argument sharing, that is, to **S** and **W**. Curry and Feys (1958) give the equivalence $\Phi = \mathbf{B}(\mathbf{BS})\mathbf{B}$, and another one necessarily involving **W**, both of which symbolize argument sharing. The correctness of syntactic types in (39) can be checked with the following derivation involving **B** and **S**.

$$(41) \quad \frac{\frac{\frac{\omega_1}{X/W/Y: f} \quad \frac{\omega_2}{Y/Z: g} \quad \frac{\omega_3}{W/Z: h}}{X/W/Z: \mathbf{B}fg}^{\mathbf{B}} \quad \frac{\omega_4}{Z: a}}{X/Z: \mathbf{S}(\mathbf{B}fg)h}^{\mathbf{S}} \\ X: \mathbf{S}(\mathbf{B}fg)ha = \mathbf{B}(\mathbf{BS})\mathbf{B}fgha = f(ga)(ha)^{\text{app}}$$

I enumerate the other arities of Φ for the record. The unary version will play a crucial role in the next chapter in radically lexicalizing coordination in all languages, where it will turn out that X , W and Y must be of the same type for this special role.

$$(42) \quad \begin{array}{ll} \text{a. } X/W/Y \quad Y/Z \quad W/Z \rightarrow X/Z & (3\Phi) \\ \text{b. } X/W/Y \quad Y/Z \rightarrow (X/Z)/(W/Z) & (2\Phi) \\ \text{c. } X/W/Y \leftrightarrow (X/Z)/(W/Z)/(Y/Z) & (1\Phi) \end{array}$$

Notice that Φ cannot be just **S**. For example, the following syntactic typing cannot be Φ , as the derivation shows. The function f is a two-argument object, not one.

$$(43) \quad \frac{\frac{\frac{\omega_1}{X/Y: f} \quad \frac{\omega_2}{Y/W/Z: g} \quad \frac{\omega_3}{W/Z: h}}{Y/Z: \mathbf{S}gh}^{\mathbf{S}} \quad \frac{\omega_4}{Z: a}}{Y: \mathbf{S}gha}^{\text{app}} \\ X: f(\mathbf{S}gha) = f(ga(ha)) \neq f(ga)(ha)^{\text{app}}$$

Now we come to a territory which even Curry and Feys (1958) find unreasonably complex and unwieldy. The combinator Ψ has the semantics

$\Psi f gab = f(ga)(gb)$. Clearly, **W** must be involved to get two g 's, and **C** must be there to account for the ordering ag . They give the following equivalence: $\Psi = \mathbf{B}(\mathbf{BW}(\mathbf{BC}))(\mathbf{BB}(\mathbf{BB}))$. We can syntacticize it accordingly.

$$(44) X/Y/Y: f \ Y/Z: g \ Z: a \ Z: b \rightarrow X: f(ga)(gb) \quad (\Psi)$$

Ψ looks artificial from a natural language perspective as well. Argument-sharing has been attested in all languages, for example *Mary wants to study*, and *John eats and Barry cooks potatoes* (whether these are done by **S**, **W** or Φ is the topic of subsequent chapters). Examples of predicate-sharing are unheard of. (This is of course not much of an explanation until we show what is odd about the syntacticized Ψ . That has to wait for another book.)

The predicate-sharing of the kind we see in gapping, for example in (45), can be conceived as *and'(like'chem'kafka')(like'eng'witt')*.

$$(45) \textit{Kafka liked chemistry, and Wittgenstein engineering.}$$

But it requires Φ semantics rather than Ψ , i.e. g and h of Φ are interpretively related in this construction rather than be identical functions, as Steedman (2000b: 188) observed.³³ The following purported syntacticization of Ψ is not valid because it fails to capture the semantic dependencies embodied in Ψ . It is inconsistent about g 's domain type.

$$(46) X/Y/Y: f \ Y/Z: g \ Z: a \ W: b \rightarrow X: f(ga)(gb) \quad (*\Psi)$$

We can also ask what is preventing (44) from receiving an interpretation such as $f(gb)(ga)$, rather than $f(ga)(gb)$ as presumed there. After all, both (ga) and (gb) are syntactically of the type Y . This is a crucial point, and it relates to our understanding of *category* as consisting of a syntactic type and a semantic type. The implication in the syntacticization (44) is that semantics of f is like (47a) below, whereas $f(gb)(ga)$ requires (47b).

$$(47) \text{ a. } X/Y/Y: \lambda p\lambda q.fpq \ Y/Z: g \ Z: a \ Z: b \rightarrow X: f(ga)(gb) \quad (\Psi)$$

$$\text{ b. } X/Y/Y: \lambda p\lambda q.fqp \ Y/Z: g \ Z: a \ Z: b \rightarrow X: f(gb)(ga)$$

$X/Y/Y: \lambda p\lambda q.fpq$ and $X/Y/Y: \lambda p\lambda q.fqp$ are not of the same category although their syntactic types are the same. Conflating the arguments to Y on the syntactic side without showing the semantic side is unhelpful in this example. I will however continue to use this practice when no confusion arises.

I enumerate the lower arities of Ψ for the sake of completeness.

$$(48) \text{ a. } X/Y/Y \ Y/Z \ Z \rightarrow X/Z \quad ({}_3\Psi)$$

$$\text{ b. } X/Y/Y \ Y/Z \rightarrow (X/Z)/Z \quad ({}_2\Psi)$$

$$c. X/Y/Y \leftrightarrow (X/Z)/Z/(Y/Z) \quad (1\Psi)$$

Next we consider Rosser's (1935) **J**, with semantics $\mathbf{J}fabc = fa(fcb)$. Like Ψ , this combinator is also predicate-sharing, which is in this case also self-embedding. **J** can be syntacticized as follows.

$$(49) X/X/Y: f Y: a X: b Y: c \rightarrow X: fa(fcb) \quad (\mathbf{J})$$

There is no language in which we have a phrase which would be in pseudo-English *John wants that Barry a book*, to mean 'John wants Barry to want a book'. The phrase would have the semantics *want'(want'book'barry')john'*, i.e. $\mathbf{J}(\mathbf{C}want')john'book'barry'$. This fact will similarly await explanation. We see no good reason to include **J** or Ψ in natural language syntax, either dependency-wise or constituency-wise, and that should do for the time being in lieu of an explanation.

Notice that for **J** both the matrix and the embedded f are syntactically two-argument functions. Note also the **C**-effect engendered by the order of the arguments X and Y , to obtain $fa(fcb)$, but not $fa(fbc)$. **J** is enumerated in lower arities below.

$$(50) a. X/X/Y Y X \rightarrow X/Y \quad (3\mathbf{J})$$

$$b. X/X/Y Y \rightarrow (X/Y)/X \quad (2\mathbf{J})$$

$$c. X/X/Y \leftrightarrow (X/Y/X)/Y \quad (1\mathbf{J})$$

We stop at this arity (as Curry and Feys 1958 did) because of two reasons: (a) Higher arities no longer add to our understanding of syntactically revealing semantic dependencies—it has already exceeded its limits in four,³⁴ and (b) we know that **S** and **K** are good enough to represent any combination, and **Y** is sufficient for recursion (but not necessary; it can be expressed in an **SK**-system albeit awkwardly).³⁵ The remaining combinators and arities are relevant to narrowing the kinds of dependencies we see in natural languages. A computer equipped with an **SK**-machine can perform any computable function just fine—see Peyton Jones (1987) for such a virtual machine.³⁶

5. Powers and combinations

The definition of powers (see the appendix) provides a natural generalization of combinators over functions of various arities. In this section we syntacticize \mathbf{B}^2 and some combinations of combinators because they are very useful in defining other generalizations.

Recall that $X^{n+1} = \mathbf{B}XX^n$, hence $\mathbf{B}^2fgab = \mathbf{B}\mathbf{B}fgab = f(gab)$. Therefore \mathbf{K}^2 deletes the two elements in \mathbf{K}^2fab , and \mathbf{S}^2 makes two copies of the third argument, rather than one copy by \mathbf{S} , because $\mathbf{S}^2fgh = \mathbf{B}\mathbf{S}\mathbf{S}fgh = f(gh)(h(gh))$.

\mathbf{B}^2 composes a two-argument function with a one-argument function. It can be syntacticized as (51).

$$(51) \quad X/Y \ Y/Z/W \ W \ Z \rightarrow X \quad (\mathbf{B}^2)$$

It will be most useful in binary and unary forms in the chapters to follow. I list them below.

$$(52) \quad \begin{array}{ll} \text{a. } X/Y \ Y/Z/W \ W \rightarrow X/Z & ({}_3\mathbf{B}^2) \\ \text{b. } X/Y \ Y/Z/W \rightarrow (X/Z)/W & ({}_2\mathbf{B}^2) \\ \text{c. } X/Y \leftrightarrow (X/Z/W)/(Y/Z/W) & ({}_1\mathbf{B}^2) \end{array}$$

Some other combinations have been found to be quite useful and thus deserve a name of their own. One source for them is referentially dependent words (pronouns), which Jacobson (1999) modeled with a combinator she called \mathbf{Z} (not to be confused with Curry and Feys's *iterator*, \mathbf{Z}_n). $\mathbf{Z}fga = f(ga)a$, hence $\mathbf{Z} = \mathbf{B}(\mathbf{B}\mathbf{W})\mathbf{B}$, as Szabolcsi (2003) noted. More simply, $\mathbf{Z} = \mathbf{B}\mathbf{S}\mathbf{C}$. We can see the $\mathbf{S}\mathbf{C}$ -effect in its syntacticization:

$$(53) \quad \begin{array}{ll} \text{a. } X/Z/Y: f \ Y/Z: g \ Z: a \rightarrow X: f(ga)a & (\mathbf{Z}) \\ \text{b. } \begin{array}{c} \omega_1 \qquad \omega_2 \qquad \omega_3 \\ \hline X/Z/Y: f \ Y/Z: g \ Z: a \\ \hline X/Y/Z: \mathbf{C}f \\ \hline X/Z: \mathbf{S}(\mathbf{C}f)g \\ \hline X: \mathbf{S}(\mathbf{C}f)ga = \mathbf{B}\mathbf{S}\mathbf{C}fga = f(ga)a \end{array} \end{array}$$

Its lower arities are listed below. ${}_1\mathbf{Z}$ is Jacobson's (1999) z . (She wrote Y/Z as Y^Z .)

$$(54) \quad \begin{array}{ll} \text{a. } X/Z/Y \ Y/Z \rightarrow X/Z & ({}_2\mathbf{Z}) \\ \text{b. } X/Z/Y \rightarrow (X/Z)/(Y/Z) & ({}_1\mathbf{Z}) \end{array}$$

Rosenbloom (1950) christened $\mathbf{B}\mathbf{B}$ with the name \mathbf{D} (Smullyan's *Dove* and Turner's 1979 \mathbf{B}'). Thus $\mathbf{D}fagb = \mathbf{B}\mathbf{B}fagb = fa(gb)$. Object g must be a function, and a, b need not be functions. We can syntacticize it accordingly:

$$(55) \quad X/Y/W: f \ W: a \ Y/V: g \ V: b \rightarrow X: fa(gb) \quad (\mathbf{D})$$

Its lower arities are listed below so that we can compare them with the unusual combinator to be tackled next. I write the results of the semantics as well in preparation of their comparison.

- (56) a. $X/Y/W: f \quad W: a \quad Y/V: g \rightarrow X/V: \lambda x.f a(gx)$ (3**D**)
 b. $X/Y/W: f \quad W: a \rightarrow (X/V)/(Y/V): \lambda g \lambda x.f a(gx)$ (2**D**)
 c. $X/Y/W: f \leftrightarrow (X/V)/(Y/V)/W: \lambda y \lambda g \lambda x.f y(gx)$ (1**D**)

Now consider **O**. Its definition is given below.³⁷

- (57) $\mathbf{O} \stackrel{\text{def}}{=} \lambda f \lambda g \lambda h.f(\lambda x.g(hx)) \quad \mathbf{O} f g h = f(\lambda x.g(hx))$ Thus $\mathbf{O} = \mathbf{CB}^2\mathbf{B}$.

The first argument of **O** is slightly unorthodox because it takes an unsaturated function as an argument. Note also that $f(\lambda x.g(hx))$ is not necessarily the same as $\lambda x.f(g(hx))$.³⁸ Therefore the syntacticized version of **O** must include an orphan argument, *Z*, as an argument of *f*, unlike **D**:

- (58) $X/(Y/Z): f \quad Y/W: g \quad W/Z: h \rightarrow X: f(\lambda x.g(hx))$ (**O**)

Syntactically, the argument types of *W/Z* and *Y/Z* above must be the same otherwise we do not capture **O**'s semantics. The following purported syntacticization is therefore wrong.

- (59) $X/(Y/V): f \quad Y/W: g \quad W/Z: h \rightarrow X: f(\lambda x.g(hx))$ (***O**)

I enumerate the lower arities of use for **O** to show that it is different than **D**; cf. (56). ${}_2\mathbf{O}$ is Hoyt and Baldridge's (2008) **D**.

- (60) a. $X/(Y/Z): f \quad Y/W: g \rightarrow X/(W/Z): \lambda h.f(\lambda x.g(hx))$ (2**O**)
 b. $X/(Y/Z): f \leftrightarrow X/(W/Z)/(Y/W): \lambda g \lambda h.f(\lambda x.g(hx))$ (1**O**)

The curious thing about **O** is that, although it is a combinator (its definition has no free variables), it is not a supercombinator, because *g* and *h* are free in its lambda-abstracted part of the body, $\lambda x.g(hx)$. Its close relative **D** is a supercombinator because its lambdas are all grouped to the left. All combinators in Table 1 are supercombinators, except **Y**. However, some expressions with inner lambdas are indeed supercombinators, for example $\lambda x \lambda y.x y(\lambda z.z)(\lambda w.0)$.

Notice that, unlike **Y**, **O** is finitely typeable. Therefore we must scrutinize it in the next chapter whether to confine **O** to the lexicon, or to let it operate freely in syntax.

Finally, consider another mixture of combinators, **BS(BB)**, equivalently **BSD**, with semantics $\mathbf{BSD} f g a b = f a(g a b)$. It is a natural generalization of

S over functions with more than one argument. (Other generalizations, such as $fa(gba)$, are already covered by **S**.) We can syntacticize it as follows.

$$(61) (X/Y)/Z: f (Y/W)/Z: g \quad Z: a \quad W: b \rightarrow X: fa(gab) \quad (\mathbf{S}'')$$

The name **S''** is suggested here to reflect its close relation to **S** and **B**². (**S'** is spoken for; it is Turner's 1979 name for **Φ**.)

The powers of **S** do not embody linguistically relevant semantic dependencies. $\mathbf{S}^2 fga = \mathbf{BSS} fga = f(ga)(a(ga))$, i.e. a is both a predicate over g and an argument of g . Likewise, powers of **C** are unhelpful. $\mathbf{C}^2 = \mathbf{BCC} = \mathbf{I}$. $\mathbf{C}^3 = \mathbf{BCC}^2 = \mathbf{C}$. However **S''** seems quite relevant. We shall see linguistic examples requiring **S''** in Chapter 5.

The crucial link in the syntactic types of **S''** is the argument types of X and Y , which must contain the same type, viz. Z , in the right order. Some purported types for f such as $(X/Y)/V$ or $(X/Z)/Y$ would not be **S''** semantics. Lower arities of **S''** materialize as follows.

$$(62) \begin{array}{ll} \text{a. } (X/Y)/Z (Y/W)/Z \quad Z \rightarrow X/W & ({}_3\mathbf{S}'') \\ \text{b. } (X/Y)/Z (Y/W)/Z \rightarrow (X/W)/Z & ({}_2\mathbf{S}'') \\ \text{c. } (X/Y)/Z \leftrightarrow (X/W/Z)/(Y/W/Z) & ({}_1\mathbf{S}'') \end{array}$$

6. Why syntacticize?

This concludes our syntacticization of the combinators. Whether combinators or supercombinators, they lend themselves to variable-free syntax in which all the semantic dependencies are imported into syntactic dependencies, and no other dependency is engendered by syntax, hence every combination is solely adjacency-based, including specification of argument-taking, i.e. lexical categories.

Schönfinkel's idea appears to be actually necessary to directly import adjacency semantics to adjacency syntax. This result was independently discovered by Curry (1929) and Ades and Steedman (1982). Chomsky (1995) has claimed that binary merge is virtually conceptually necessary. (Unary *move* is considered virtually conceptually necessary as well, in Chomsky 2005, which is related to Schönfinkel's **T**.) We now know that they are not. **T** follows from **S** and **K**. Binary merge follows from currying, which is a theorem. The theorem crucially relies on the prefixed binary juxtaposition of Schönfinkel. Therefore, Chomsky is right to claim that it is a conceptual necessity, if we

take that to mean a theoretical necessity, but wrong to dismiss a need for scientific justification of it. Combinators show how we can justify it.

The discussion in this chapter might have given the impression that the practice expounded here is to promote the meaning-to-form direction of translating semantic types to syntactic types, as opposed to form-to-meaning translation of for example Chomsky (1970), where the X-bar theory of phrase structure is mapped onto meanings, or the Klein and Sag (1985) model, where syntactic categories and phrase structure rules are translated into semantic types.

This is not the case. The ‘:’ notation embodies lexical codetermination rather than determination. It is a radical lexicalization and combinatorization of Bach’s (1976) *rule-to-rule hypothesis*, by which, rather than Montague-Bach-style rules, which would make us worry about whether the syntactic one or the semantic one is the determinant, we only have words with combinatory categories. By their very nature, they need to be specified uniquely. Thus the discussion of priority of syntactic rules and semantic rules becomes moot.

The reason for going through the trouble of syntacticizing the combinators is worth reiterating: they work on semantic objects, functions if you like, whereas human language observables are syntactic objects, namely words. Of course there can be other ways to go from semantics to syntax or from syntax to semantics. The combinatory theory suggests that adjacency is all we need.

The point of importing all semantic dependencies to syntax and creating no extra ones is to obtain a purely syntactic type-driven syntax. This aspect is the main source of confusion in analogies to form-to-meaning and meaning-to-form approaches. Like all analogies including mine in the preface, it is misleading, and obscures the true nature of what combinatory syntax does: it gives us compositional semantics for free, and in lock-step with syntax, i.e. incrementally. The talk of having “a semantically motivated grammar” in correspondence theories to hint at the psycholinguistic plausibility (e.g. left-to-right processing) is unhelpful because there can be no semantically unmotivated grammar. A grammar without semantics is no grammar.

The combinators covered so far seem to be deterministically translatable to syntactic types, but they were designed to be that way to begin with. Radical lexicalism predicts that natural language is one domain in which one-way determinism cannot hold for all compositional meanings. It does depend on the word, and the possible languages we get out of these singularities do not differ in arbitrary ways, due to adjacency being the only primitive on which

multiple constraints on language can act, for example the constraints which manifest themselves in the knowledge of words including predicate-argument structure, constituent structure, information structure and intonational structure.

It will turn out that most of the syntactic manifestations of combinators, and most of the combinators, are only relevant to the lexical items, not to the freely operating universal rules. I took pains to enumerate them in all arities so that we can compare the alternatives from a linguistic perspective. This requires a set of substantive principles to choose which ones go to the lexicon and which ones stay as freely operating. This is the topic of the next chapter.

Chapter 5

Combinatory Categorical Grammar

Mark Steedman's Combinatory Categorical Grammar, CCG, is a theory of syntax-semantics for natural languages in which only the combinators that directly and solely bear on constituency operate in syntax freely, all others being radically lexicalized.³⁹ His conjecture so far has been that this is a **BTS** system. Free operation arises from noninterdefinability. His counteracting force for this theoretical result is the empirical test of constituency. No combinator which is syntacticized can do the work of others, and its syntactic work cannot be done by others.⁴⁰

CCG is strictly Schönfinkelian because the only primitives of the system are forward and backward application, which are the syntacticized versions of Schönfinkel's juxtaposition. All lexical functions are curried, all syntactic rules arise from combinators, and every principal functor in syntax schematized below faces only one adjacent syntactic object:

$$(1) \begin{array}{l} X/\cdots \quad \cdots \rightarrow X\cdots \\ \cdots \quad X\backslash\cdots \rightarrow X\cdots \end{array}$$

X is called the *principal functor*. The result type of the binary combination is uniquely determined by X . This is semantic in origin (but clearly syntacticized), because it amounts to saying that X is the projected result type in the local configuration of (1). Because this result arises from the semantics and syntax of combinators as shown in the previous chapter, CCG does not need an extraneous projection principle; it is predicted by the type-dependence of radically lexicalized natural language grammars.

1. Combinators and wrapping

By definition, any system that employs surface wrap ceases to be a combinator system, because no combinator can do the work of wrap, and if we assume that a syntacticized rule does the work of wrap, no combinator can match it on the semantic side. We would lose the combinatory base of directly and immediately associating an interpretation with every syntactically combinable constituent.

This result might be puzzling at first, knowing that **C** does the equivalent of wrap, because $\mathbf{C}abc = acb$. However, this behavior presumes a wrap interpretation only if we think of ab as a holistic unit in syntax or semantics, which is split by c by being wrapped in them (it is also commonly referred to as “ ab wraps around c ”). The syntacticization of **C**, repeated below, made no such assumptions. Y and Z are categories of independent syntactic objects. It would not matter whether we binarize the rule as in the second line. The string-view of **C** is provided in (2c), in preparation of its comparison with wrap.

$$\begin{array}{ll}
 (2) \text{ a. } (X/Y)/Z: a \ Y: b \ Z: c \rightarrow X: acb & (\mathbf{C}) \\
 \text{ b. } (X/Y)/Z: a \ Y: b \rightarrow X/Z: \lambda c.acb \\
 \text{ c. } \frac{\frac{s_1}{X/Y/Z: a} \quad \frac{s_2}{Y: b} \quad \frac{s_3}{Z: c}}{s_1 s_2 s_3 := X: acb}^c
 \end{array}$$

We must distinguish systems with **C**, which are combinatory, from systems with wrap, which are not. So what exactly is syntactic wrap, and why is wrap not so subversive when done lexically or semantically? Here we must look to Bach (1980, 1984), Dowty (1996).⁴¹ Below is Bach’s (1984) syntactic formulation of wrap translated to current notation. The slash is modalized to wrap, following Jacobson (1992).

$$(3) \frac{\frac{s_1}{X/_W Y: a} \quad \frac{s_2}{Y: b}}{\text{first}(s_1) s_2 \text{rest}(s_1) := X: ab}^{\text{wrap}} \quad (\text{wrap})$$

where $\text{first}(x)$ means the first element of a list of structures for Bach (first word for Dowty 1996), and $\text{rest}(x)$ means the remainder.

Notice that, semantically speaking, wrap is application, whereas surface-syntactically there is no combinatory counterpart. Naturally, this cannot be **C**. Observe also Bach’s derivation of *persuade John to do the dishes* as surface wrap:⁴²

$$(4) \frac{\frac{\frac{\text{persuade}}{(S \setminus NP) /_W NP / VP} \quad \frac{\text{to do the dishes}}{VP} \quad \frac{\text{John}}{NP}}{\text{persuade to do the dishes} := (S \setminus NP) /_W NP} \rightarrow}{\text{persuade John to do the dishes} := (S \setminus NP)}^{\text{wrap}}$$

Let us now consider Dowty's examples for wrap, the resultatives and verb-particle pairs: *hammer (the metal) flat*, *let (the dog) loose*, *look (the word) up*, where discontinuity is shown by parentheses. As he points out, *hammer round* does not have the same behavior as *hammer flat*, therefore we must assume *hammer flat* as a lexical item, which necessarily wraps.

The implicit assumption here is that the meaning of *hammer flat* is something like *hammerflat*,¹ not *hammer'flat*.¹ The application of *hammerflat*' to *metal*' gives us *hammerflat'metal*,¹ stringwise *hammer the metal flat*, following (3), but not (2). This is indeed wrap in the noncombinatory sense because no combinator can split *hammerflat*' into pieces, whereas **C** can do that to the sequence *hammer'flat*' easily. Similarly, *look up* as a lexical entry can be *lookup*,¹ or *look'up*.¹ In the first case, there is no combinator to get *look the word up*, hence a combinatory system must assume two semantic objects *look*' and *up*,¹ whereas a wrap system (of type-dependent or structure-dependent variety) would have more degrees of freedom in lexical options.

Dowty extends this view to phrasal items and the Wackernagel position (the second position which clitics universally tend to attach themselves phonologically), to the so-called nonadjacent phenomena in languages. It was also the motivation in Bach (1984) to analyze *persuade John to do the dishes* as the wrap of *John*, a syntactically and semantically independent object, inside *persuade to do the dishes*.

This move reintroduces **C** in addition to wrap for the reasons just discussed: we must assume that the dependencies arise from *Cpersuade'tdtd'john*,¹ because they are syntactic phrases. This is the motive for '/*w*' in (4), which turns everything into function application semantics, i.e. *persuade'john'tdtd*.¹ The surface combination, however, is not **C**.

Bach's formulation of wrap is independent of phonology, but Dowty's interpretation of it is morphophonological, because it assumes that wrap knows word boundaries. In any case, an "infix here" point must be remembered for every lexical item, which must be maintained properly throughout phrase combination, and herein lies another problem. In languages where the notion of word is linear-recursive (such as Turkish and Gusii; see Hankamer 1989, Creider, Hankamer and Wood 1995), this seems to require a finite-state machine running through word boundaries during the syntactic process, in addition to the syntax-phonology interface with its own computations of exactly the same nature.

The Wackernagel phenomenon below forces this assumption, where the focus- and coordination-clitic *de* necessarily wraps into the second conjunct

with a recursive first word. This phenomena and its related wrap behavior must be explained, rather than assumed as knowledge to go with a lexical slash such as ‘/w’.

- (5) *Mehmet bugün gelecek, Ev-de-ki-nin-ki-ler de yarın.*
 M today come-FUT house-LOC-ki-POSS-ki-PLU FOC tomorrow
 lit. ‘Mehmet is coming today, and the ones of who is in the house to-
 morrow’
 meaning, e.g. ‘The family of the girlfriend of the boy in the house will
 come tomorrow.’ Turkish

The semantic (therefore lexical) use of wrap does not threaten the combinatory base of CCG, because it amounts to a local use of **C** rather than wrap. It has been employed by Szabolcsi (1989) and Steedman (2000a) to handle for example ditransitive constructions and VSO languages.

Examples (6a–b) are from Szabolcsi, where she assumes for reasons cited in the paper the category $(S\backslash NP)/NP/PP$ for *introduce*, rather than the surface word order $(S\backslash NP)/PP/NP$. Then, because of (6c), we must apply unary **B** to $VP\backslash(VP/PP)$ to get $(VP/NP)\backslash(VP/PP/NP)$ first, in the lexicon, and apply unary **C**, again in the lexicon, to simulate wrap, which yields lexically the category $(VP/NP)\backslash(VP/NP/PP)$.

- (6) a. *John introduced Mary to himself* Szabolcsi (1989: 307)

$$\frac{(S\backslash NP)/NP/PP \quad VP\backslash(VP/PP)}{VP\backslash(VP/PP)}$$
 b. *John introduced Mary to herself*

$$\frac{\frac{(VP/NP)\backslash(VP/PP/NP)^{\text{lex B}}}{(VP/NP)\backslash(VP/NP/PP)^{\text{lex C}}}}{VP\backslash(VP/PP)}$$
 c. *John introduced Mary to himself and Susan to herself.*

This way we maintain a type-raised syntactic object in all cases including reflexives, which is an important part of Szabolcsi’s organization of grammar.

Steedman (1996b, 2000a) puts LF to work in (6). I compare the three CCG proposals for LF phenomena in Chapter 6. His suggestion for VSO languages is a category such as (7a) for Welsh, because of (7b–c). This is also lexical/semantic wrap, not syntactic, because the lambda term of the verb is **Cverb**’.

- (7) a. VSO verb := $S/NP/NP_{\text{agr}}$: $\lambda x_1 \lambda x_2 . \text{verb}' x_2 x_1$

b. *Gwelodd Wyn ef ei hun*
 Saw Wyn himself
 ‘Wyn saw himself.’

Awbery (1976: 131)

c. **Gwelodd ef ei hun Wyn*
 Saw himself Wyn

The treatment of adjacency creates two worlds for combinatory linguistic categories, one in which adjacency as the sole base looks at possible categories (i.e. possible languages) by enumerating all adjacency-based categories, and the other in which adjacency effects and other factors are incorporated into theories as needed (e.g. Moortgat and Oehrle 1994). Because of the mediating subtheories in the latter kind of framework and the use of a logical form in the first one, cotranslatability of the categories in the two categorial worlds is becoming increasingly difficult. One can see the clear split in Combinatory Categorial Grammars and Type-Logical Grammars, although there are many points of contact and good sources of inspiration both ways (cf. Morrill 1994, Moortgat 1988b, Carpenter 1997, Moortgat and Oehrle 1994, Baldrige 2002, Kruijff and Baldrige 2004, Hoyt 2006).⁴³

2. Linguistic categories

What are the solely adjacency-based categories *for language*? The crux of the matter is that whatever the nature of these categories is, they are the categories *of* syntactic objects. This is an empirical requirement, because the observables are the syntactic objects, namely words, not the semantic objects. A category is a hypothesis about what the syntax-semantics connection of the observables could be. That of course does not prevent categories from being semantic in nature, as Edmund Husserl (1900) claimed to be the case:

Clearly we may say that if presentations, expressible thoughts of any sort whatever, are to have their faithful reflections in the sphere of meaning-intentions, then there must be a semantic form which corresponds to each presentational form. This is in fact an *a priori* truth. And if the verbal resources of language are to be a faithful mirror of all meanings possible *a priori*, then language must have grammatical forms at its disposal which give distinct expression, i.e. sensibly distinct symbolization, to all distinguishable meaning-forms.

Logical Investigations vol.II: 55

Many categorial grammarians consider Husserl’s statement to be the birth of categorial grammar. This is not surprising, because of the implicit com-

mitment since the early years of categorial grammar to have the substantive categories associate only with verbal resources, namely words (as opposed to say with both words and grammar rules). This is an explicit commitment in Combinatory Categorical Grammar:

(8) *Radical Lexicalism:*

All language-particular information is in the lexicon.

The term *Radical lexicalism* is due to Lauri Karttunen (1989). The method is described in the appendix. Radical lexicalism in light of Husserl's desiderata suggests two manifestations of categories: formal categories and substantive categories. Formal categories are universal generalizations of the substantive categories, hence they are not different in kind.⁴⁴ We can think of the syntacticization of combinators as yielding formal categories, for example $X/Y: f$, $X: fa$ and $Y: a$ below for application.

(9) $X/Y: f \quad Y: a \rightarrow X: fa$

Any substantive category can substitute for X and Y above (provided that the desired adjacency configuration required by the rule is satisfied). This is not true of substantive categories, say $S \backslash NP$, where S means "sentence" and NP means 'noun phrase'. Only NPs can substitute for NP , to get *Kafka*, or *The stories of Poe* etc.

Similarly, semantically open propositions can be substituted for S/NP to get *The man devoured*, or *Mary hit*, but not **Kafka chemistry* where an object of category NP (*Kafka*) attempts to substitute for S/NP . Note that the sequence *Kafka chemistry* can be predicational, but this interpretation is parasitic on a verb, as in gapping:

(10) *Wittgenstein adored engineering, and Kafka chemistry.*

Here, the required category is not S/NP for *Kafka*. It is NP , as the semantics of the sentence proves.

To be able to distinguish *Wittgenstein adored* from *adored Wittgenstein* in the Husserlian sense, we must categorize them differently although both are open propositions semantically. In CCG parlance, the former is S/NP and the latter is $S \backslash NP$. The difference in slashes is a forced move of syntacticization.

Unlike the semantically-motivated combinators in which we can choose to represent all functions in prefix notation, the syntactic objects of languages vary in directionality. Tagalog is head-initial whereas Turkish is head-final. English is head-initial (e.g. *of the book*) and head-medial, as in its basic word

order SVO. Thus the syntacticization of combinators must consider this aspect as well to complete the picture.

We can think of ‘backward application’ as the only other possibility of application because there is only one function and one argument in application, i.e. only one slash:

$$(11) Y : a \quad X \backslash Y : f \rightarrow X : fa \quad (<)$$

The semantic dependencies of application are preserved in this version as well; the semantic result is fa , not af . This factoring of order into the categories is reflected in the name of the rule, viz. ‘<’ for backward application and ‘>’ for forward application.

Thus the following purported manifestations of application are ruled out because they do not preserve the semantic dependency instigated by order:

$$(12) \text{ a. } X \backslash Y : f \quad Y : a \rightarrow X : fa \quad (*>)$$

$$\text{ b. } Y : a \quad X / Y : f \rightarrow X : fa \quad (*<)$$

In a configuration where there is more than one slash, for example in the binarized composition (13a), the possibilities in (13b–d) preserve the semantic dependency of order, but the ones in (13e–h) do not. Thus we can subsume Steedman’s (2000b) principles of consistency and inheritance, which helped to eliminate configurations such as (13e–h), by the semantics of order inherent in combinators.⁴⁵

$$(13) \text{ a. } X / Y : f \quad Y / Z : g \rightarrow X / Z : \mathbf{B}fg \quad (> \mathbf{B})$$

$$\text{ b. } Y \backslash Z : g \quad X \backslash Y : f \rightarrow X \backslash Z : \mathbf{B}fg \quad (< \mathbf{B})$$

$$\text{ c. } X / Y : f \quad Y \backslash Z : g \rightarrow X \backslash Z : \mathbf{B}fg \quad (> \mathbf{B}_\times)$$

$$\text{ d. } Y / Z : g \quad X \backslash Y : f \rightarrow X / Z : \mathbf{B}fg \quad (< \mathbf{B}_\times)$$

$$\text{ e. } X / Y : f \quad Y \backslash Z : g \rightarrow X / Z : \mathbf{B}fg \quad (*> \mathbf{B}_\times)$$

$$\text{ f. } Y / Z : g \quad X \backslash Y : f \rightarrow X \backslash Z : \mathbf{B}fg \quad (*< \mathbf{B}_\times)$$

$$\text{ g. } X \backslash Y : f \quad Y / Z : g \rightarrow X / Z : \mathbf{B}fg \quad (*> \mathbf{B}_\times)$$

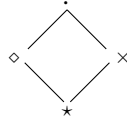
$$\text{ h. } Y \backslash Z : g \quad X / Y : f \rightarrow X \backslash Z : \mathbf{B}fg \quad (*< \mathbf{B}_\times)$$

These restrictions are forced moves in the theory. In (13e–f), the directionality of Y is respected but the directionality of Z is not. In (13g–h), the directionality of Z is respected but the directionality of Y is not. All directionalities are respected in (13a–d). Notice that directionality is inherently a syntactic property of the argument and not the result, as first observed by Steedman (1991b). Thus there is no directionality of X above, and all directionalities are accounted for in the logic of the argument from order semantics.

A backward or forward slash is not necessarily a crossing slash. This information needs to be contextualized, for example as $'/_x'$ for a crossing forward slash and $'/_\diamond'$ for a harmonic forward slash. A “don’t care” forward slash can be contextualized too, as $'/_.'$. These aspects are relevant in contexts in which the curried binary configuration involves two or more slashes, as above. Since there is one slash in application, we can make categories application-only too, with the most restrictive slash $'/_*$ ' (likewise for the backward slash). In a purely applicative system, these modalities exhaust the possibilities for slash contextualization.

These are the modalized combinatory categories of Baldrige (2002). He defines the following hierarchy as a way of compiling the knowledge of slash compatibility:

- (14) CCG type lattice for slash modalities (from Baldrige and Kruijff 2003):



The dot is the least restrictive modality, the star the most restrictive. The diamond and the cross are partially restrictive and mutually incompatible. Thus a $'/_*$ ' slash is only compatible with itself, and $'/_.'$ is compatible with all forward slashes (similarly for backward slash). The least restrictive modality is omitted by convention to avoid further notational clutter.

- (15) The $'/_\backslash'$ is same as $'/_\backslash.'$. The $'/_/'$ is same as $'/_/.'$. (dot omission)

Now we can refine the syntacticized combinators of this section:

- (16) a. $X/_*Y : f \quad Y : a \rightarrow X : fa$ ($>$)
 b. $Y : a \quad X/_*Y : f \rightarrow X : fa$ ($<$)
 c. $X/_\diamond Y : f \quad Y/_\diamond Z : g \rightarrow X/_\diamond Z : \mathbf{B}fg$ ($> \mathbf{B}$)
 d. $Y/_\diamond Z : g \quad X/_\diamond Y : f \rightarrow X/_\diamond Z : \mathbf{B}fg$ ($< \mathbf{B}$)
 e. $X/_x Y : f \quad Y/_x Z : g \rightarrow X/_x Z : \mathbf{B}fg$ ($> \mathbf{B}_x$)
 f. $Y/_x Z : g \quad X/_x Y : f \rightarrow X/_x Z : \mathbf{B}fg$ ($< \mathbf{B}_x$)

The goal of introducing the combinatory modalities is to make finer distinctions in the Husserlian sense, for example to distinguish $S/_\diamond NP$ of *Wittgenstein would adore* from $S/_x NP$. The need to distinguish these categories is forced by the data, under the assumption of adjacency-only syntax:

(17) *The field which I think that Wittgenstein would adore is web engineering.*

We are forced by related data to categorize *that* as $S' / \diamond S_{\text{fin}}$:

(18) **The philosopher who I think that would adore Wittgenstein is Russell.*

This category disallows the combination of *that* with *would adore Wittgenstein*, which is the critical difference between (17) and (18). The newly introduced syntacticization in (16) would be in vain if we could not distinguish **that would adore Wittgenstein* from *that Wittgenstein would adore*. The critical steps of (17) and (18) are shown below, in which the differing possibility of (16c) versus (16e) does the critical work. Thus we must distinguish $S' / \diamond S_{\text{fin}}$ from $S' / \times S_{\text{fin}}$, the latter of which would allow (19b).

- (19) a. *that Wittgenstein would adore*

$$\frac{S' / \diamond S_{\text{fin}} \quad S_{\text{fin}} / \diamond NP}{S' / \diamond NP} \rightarrow \mathbf{B}$$
 b. *that would adore Wittgenstein*

$$\frac{S' / \diamond S_{\text{fin}} \quad S_{\text{fin}} \setminus NP}{***} \rightarrow \mathbf{B}_\times$$

The star modality is also a forced move, given the adjacency assumption and Husserl's desiderata. *And*'s category must be more refined than $(S \setminus \diamond S) / \diamond S$ and $(S \setminus \times S) / \times S$:⁴⁶

- (20) a. **player that shoots and he misses*

$$\frac{(N \setminus \diamond N) / \diamond (S \setminus NP) \quad S \setminus \diamond NP \quad (S \setminus \diamond S) / \diamond S \quad S}{S \setminus \diamond S} \rightarrow$$

$$\frac{S \setminus \diamond NP}{\mathbf{B}}$$
 (Baldrige 2002)
 b. **Kafka and he studied chemistry smiled.*

$$\frac{S / i (S \setminus i NP) \quad (S \setminus \times S) / \times S \quad S \quad S \setminus NP}{S \setminus \times S} \rightarrow$$

$$\frac{S / \times (S \setminus \times NP)}{\mathbf{B}_\times} \rightarrow$$

$$S$$

Under the present method of syntacticization without extra assumptions over and above adjacency, both examples would be fine if we did not have $(S \setminus \times S) / \times S$ for *and*, and did not work with the modalized combinators of (16).

For example, (13f) would allow (20b) as shown in the derivation. The examples in (20) also demonstrate a convenient generalization of directionality: the underspecified slash.

- (21) ‘| m ’ stands for ‘\ m ’ and ‘/ m ’, with modality m . (dir. underspecification) m can be underspecified too, as in (20b)’s *Kafka*.

We can now distinguish the relative pronouns *that* and *whom* by typing them with the categories $(N \setminus_{\diamond} N) /_{\diamond} (S \setminus NP)$ and $(N \setminus_{\diamond} N) /_{\diamond} (S / NP)$, respectively:

- (22) a. *the field that* [*Kafka admired*] $_{S \setminus NP}$
 b. *the field that* [*admired Kafka*] $_{S \setminus NP}$
 c. **the chemist* *whom* *admired Kafka*

$$\frac{(N \setminus_{\diamond} N) /_{\diamond} (S / NP) \quad S \setminus NP}{***}$$

The last bit of differentiation to make good on Husserlian categorization is the difference in *like* versus *likes*. As these are related but different words (in fact, the same lexeme is involved), we would expect their categories to be related but different. Following Kay (1985), Shieber (1986), we decorate the basic (nonslashed) categories with features. We abbreviate them to save space; 3s is short for AGR=3s, where AGR is an agreement feature.

- (23) *likes* := $(S \setminus NP_{3s}) / NP$
like := $(S \setminus NP_{-3s}) / NP$

The feature geometry can in principle be language-particular, and need not concern us here.⁴⁷ We shall however make use of common generalizations such as AGR and FIN(inite). Suffice it to say that we do not need a sophisticated theory such as that of Gazdar et al. (1985), Pollard and Sag (1994), Calder, Klein and Zeevat (1988) in which unification does nontrivial linguistic work, whereas the working hypothesis of the book is to let syntacticized combinators do all the work except basic category matching.

The theory also differs from Chomsky (1995) where a universal feature geometry is attempted. Features can be radically lexicalized just like combinatory categories. The process might miss some early generalizations over categories and features, but so be it. The generalizations that will arise from order semantics is our present concern. We can attempt to recapture the same generalizations, and hopefully more, after we flesh out all attested linguistic categories. One such example is the reworking of functional features as combinatory categories, which we do in §9.7.

To summarize, the following is the landscape of the syntactic types.

- (24) Take F to be a feature geometry (a finite set of features).
 Let $V \subseteq F^{v(V)}$ be a set of valuations of features from some value space V mapped by v .
 Take B to be a finite set of basic categories (without slashes).
 Let $S = B^V$. (All possible feature-decorated basic categories)
 Let $M = \{ \cdot, \diamond, \times, * \}$. (The set of modalities)
 Define C (the set of possible syntactic types):
 Any member of S is a potential type in C .
 If $A \in C$ and $B \in C$, then $A \mid_m B \in C$, for some $m \in M$, and $\mid \in \{ \backslash, / \}$.
 $A^B \in C$ if $A \in C, B \in C$.
 Nothing else is in C .

Explicitly enumerating the countably infinitely many distinguishable categories is the starting point of sieving some of the categories as unlikely categories for human languages.

Naturally, *Kafka's* category NP is not discriminating enough, thus we can write $NP: kafka'$ to distinguish it from $NP:wittgenstein.'$ Such obvious distinctions will be abbreviated for the sake of exposition.

The exponent category A^B semantically denotes a function from B to A , and differs from $A \mid B$ because it does not introduce a syntactic function. It is the main syntactic source for Jacobson 1999-style combinatory referential dependencies, and it has predictive powers in that field, for example relating the extraction domain $(N \setminus N) / (S \mid NP)$ to the relativization domain $(N \setminus N) / S^{NP}$ of resumptive pronouns. Its use in CCG so far has been constrained to cases where B is a basic category.

One final constraint on lexical syntactic types relates to lexical generalizations, where we can refer to a set of types and pick the ones that satisfy a constraint. It is the dollar convention of Steedman (2000b).

- (25) $T\$A$ stands for the finite set of categories TA (\$-convention)
 such that functions in T are lexical and onto T .
 A can be empty.
 $T\$A$ is empty if T is empty.

For example, $S\$$ for Turkish is $\{S, S \setminus NP, S \setminus NP \setminus NP, S / (S \setminus NP), \dots\}$. The set $S / NP\$$ would be empty. $S \setminus \$NP$ for English is the set $\{S \setminus NP, (S \setminus NP) / NP, (S \setminus NP) / NP / NP, \dots\}$. Categories $(S \setminus NP) / PP$ and S / NP are excluded.

The claim of CCG is that a grammar solely consists of radically lexicalized categories, lexically pairing the combinatory syntactic types decorated with features with a PADS, per word. Any context-free phrase-structure grammar and linear-indexed grammar can be reduced to its lexicon if we are willing to translate distributional categories such as N, V, A, P to combinatory categories. A category *is* a rule as an intensional device. Practicing linguistics “without rules” and “with principles” does not change the operative maxim. Hence Bach’s rule-to-rule hypothesis is relevant to any linguistic theory that makes use of the notion of computation and Turing representability where the notion of “rule” is built-in.

This brings us back to the troublesome interaction of feature spaces, rules and mappings. All mappings leak, unless they are lexical. Even then they are underdetermined by external meanings, which is why some statistical book-keeping must be connected to the use of a lexical correspondence. Radical lexicalization adds to this observation the property that if we radically lexicalize all structure-building, then one end of the mapping or rule ought to be some kind of compositional semantics (logical form, predicate-argument structure, dependency relations, etc.). Radical lexicalism in this narrow sense goes back further than Karttunen (1989), who coined the name. In the famous 1960 conference which also included contributions by Chomsky and replies to and from his critics, Lambek (1961: 169) expressed the program:

For our purpose it will be convenient to think of a phrase structure grammar as follows: the dictionary assigns to each atomic phrase a finite number of primitive types. The grammar consists of a finite number of rules of the form $p_i p_j \rightarrow p_k$ where the p_i are primitive types.[fn]

While it seems unlikely that the elimination of grammatical rules in favor of dictionary entries can be carried out for every phrase structure grammar in this sense (without making the dictionary infinite), this can be done in many examples (in fact all that I have tried).

His following suggestion can be taken as the start of the program: “It may happen that type assignments in a dictionary entry are in a sense stronger than the explicit rules of a phrase structure grammar” Lambek (1961: 170), which he illustrates in the remainder of the paper using pronouns and *wh*-items.⁴⁸

3. CCG is nearly context-free

The inadequacy of categorial grammars might have been thought to be true in 1964—with the crucial exception of Lambek (1958, 1961). It became doubtful by the publication of Geach (1972), Shaumyan (1977), Ades and Steedman (1982), Joshi (1985) and Oehrle, Bach and Wheeler eds. (1985/1988), and proven to be wrong by Joshi, Vijay-Shanker and Weir (1991), Vijay-Shanker and Weir (1994).

The emerging formal class of languages, which Aravind Joshi named *mildly context-sensitive languages* (MCSL) in its upper limit, are a superclass of context-free languages and subclass of context-sensitive languages, with a well-defined algorithmic substrate (embedded push-down automata).

The least powerful extension of context-freeness is achieved by linear-indexed grammars (Gazdar 1988), which characterize Linear-indexed Languages (LILs). Lexicalized tree-adjoining grammars (LTAG; Joshi and Schabes 1992) and CCG are provably linear-indexed (Joshi, Vijay-Shanker and Weir 1991). The desirable features include (a) polynomial-time parsability and (b) the constant-growth property of MCSLs, which ensures that all the languages of this class have strings whose lengths grow linearly, and (c) efficient parsability. Although all MCSLs are polynomially parsable, they are not all efficiently parsable, which LILs are. That is why they are the computationalists' choice of algorithmic substrate when full coverage of nested and crossing dependencies is attempted. An example of the latter is shown below.

- (26) ..omdat ik_1 Cecilia $_2$ de nijlpaarden $_3$ zag $_1$ voeren $_{2,3}$ Dutch
 ..because I Cecilia the hippopotamuses saw feed
 ‘..because I saw Cecilia feed the hippopotamuses.’

We know for example that Shieber's (1985) Swiss German data and Huybregts's (1976) Dutch data such as above are provably above context-freeness, and properly within the class of nearly context-free languages, for there are LTAG and CCG grammars for them.

Vijay-Shanker and Weir (1994) and Joshi, Vijay-Shanker and Weir (1991) proved and exemplified that for every combinatory categorial grammar, there is a linear-indexed grammar and vice versa. These grammars have nonterminals which can be associated with a stack, and the stack can be passed from/to the left nonterminal to/from a single nonterminal on the right-hand side of a rule, which restores our problem of radically lexicalizing CCG grammars because it suffices to have a single symbol on the left-hand side of every rule.

Translation to CCG is roughly as follows: CCG categories can be viewed as their result category plus a stack-valued feature identifying their arguments and the order of their combination. For example, NP is $NP[]$, and $S \backslash NP_a / NP_b$ is $S[NP_a, NP_b]$ in the stack-equipped nonterminals of a linear-indexed grammar. The ‘ $/NP_b$ ’ must be on top of the stack because it is the first argument to combine in the CCG category. Thus the stack preserves the relative order and currying of the CCG category.

The linear order of arguments for example in ‘ $S \backslash NP_a / NP_b$ ’ is encoded in the grammar rule, not in the stack. In this case the linear-indexed rule would be $S[.] \rightarrow NP_a V NP_b$, if we think of $S[., NP_a, NP_b]$ as V ’s category. Since every linear-indexed language has a linear-indexed grammar, radical lexicalization up to and including Dutch and Swiss German crossing dependencies is complete.

I show CCG’s handling of the Swiss German crossing dependencies in Figure 5. The indices in the figure are meant to facilitate to trace the derivation of correct semantics. Steedman (2000b) shows the Dutch case. I chose Swiss German because the requirements seem more strict on the syntactic and the semantic side. All arguments in a subordinate clause are case-marked in Swiss German, and they must match the subordinate verbs’ case requirements; see Shieber (1985) for discussion. The derivation’s mechanism is the topic of the next section.⁴⁹

4. Invariants of natural language combination

CCG claims that there are two kinds of semantic dependencies which have a direct reflection on syntactic processes: invariants, which need not be stipulated in the grammar of every language (the so-called universal dependencies), and lexicalizable dependencies that need to be part of a language’s grammar. The syntacticization of semantic dependencies by combinators serves both resources, thus we need empirical and theoretical grounds to decide whether a dependency is lexicalizable or not, and whether it should be lexicalized if it is lexicalizable.

An example of forced lexicalization is Inuit’s constraint that ergative NPs cannot be relativized (Manning 1996). This is something the head of relativization must enforce, say by requiring a domain of type $S \backslash NP_{abs}$, because the language is verb-final and relatively free word order, hence an extraction domain such as $S \backslash NP$ is clearly possible but not opted for by Inuit. It has

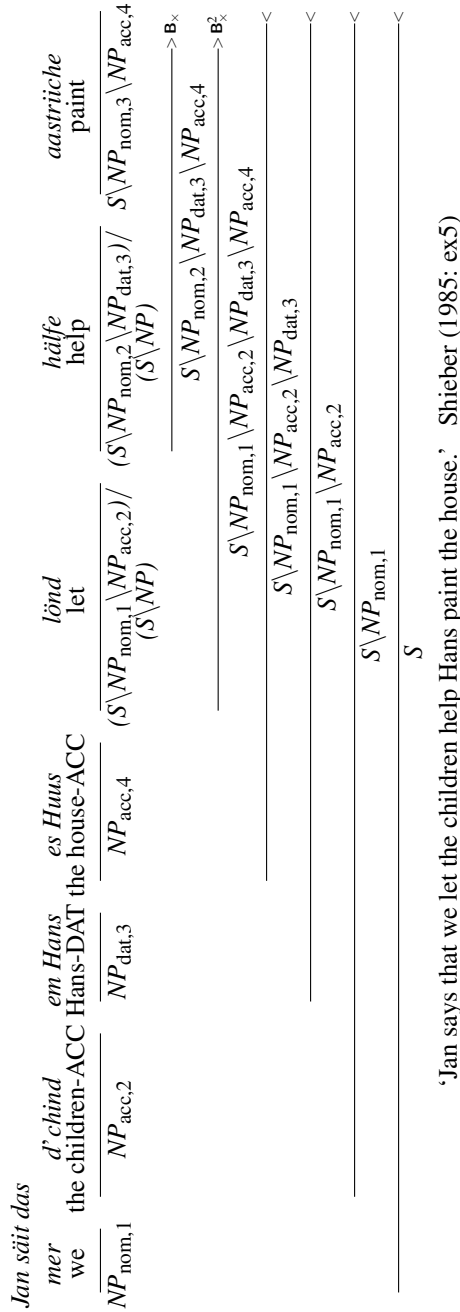


Figure 5. Swiss German crossing dependencies in CCG.

transitive participial forms, which could easily allow ergative NP extraction if not constrained in the grammar of Inuit.

We can think of Ross's (1967) Coordinate Structure Constraint, its exceptions and exceptions to exceptions, as examples of global asymmetries captured by invariants without further assumption in the lexicalized grammar, as shown in §2(14). No special constraint is needed to capture these properties. The lexical constraint on the coordinator, that it requires like-categories to maintain the semantics of coordination, is motivated independently of extractability and nonextractability. There will be no freely operating "relativization combinator" or "coordination combinator," and we would expect constituents that undergo these constructions to be quite opaque to the lexically licensed meanings of relative markers and coordinators.

The notions of redundancy and opaqueness to syntactic processes, therefore flexible constituency, play a decisive role in determining the invariants. As the discussion in Chapter 4 implied, the kind of work that ternary and quaternary combinators do at their defined arities can be done by lower arities and application. This was shown for ternary **B**, **C** and **S** syntactically. Similar results await quaternary combinators. Since we know from Schönfinkel's original work that **S** and **K** are good enough to capture all effectively computable dependencies (and more), the faithful syntacticization of the combinators without extra assumptions suggests that the same holds for the syntactic variety of other combinators.

S is ternary and **K** is binary, and we know that ternary **S** is redundant if we have binary **B**, unary **W** and unary **C**. I repeat this result here, from §4(37):

$$(27) \quad \frac{\frac{\omega_1}{(X/Y)/Z: f} \quad \frac{\omega_2}{Y/Z: g} \quad \frac{\omega_3}{Z: a}}{X: fa(ga)} \quad \frac{\frac{\omega_1}{(X/Y)/Z: f} \quad \frac{\omega_2}{Y/Z: g} \quad \frac{\omega_3}{Z: a}}{\frac{(X/Z)/Y: Cf}{\frac{X/Z/Z: \mathbf{B}(Cf)g}{\frac{X/Z: \mathbf{W}(\mathbf{B}(Cf)g)}}{\mathbf{B}}} \text{app}} \text{app}$$

Binary **B** is indispensable for purely adjacency-based solutions to examples such as (28a). The critical point of the derivation is shown in (28b). It is also justified by the constituent behavior of the same substring, for example *Who do you believe that Mary likes and John detests?*

$$(28) \quad \text{a. } \textit{Who do you believe that Mary likes?} \quad \text{Szabolcsi (1989)}$$

$$\begin{array}{c}
 \text{b. } \textit{Who do you believe that Mary likes?} \\
 \hline
 \text{NP} \quad (\text{S}\backslash\text{NP})/\text{NP} \\
 \hline
 \text{S}/(\text{S}\backslash\text{NP})^{\text{T}} \\
 \hline
 \text{S/NP}^{\text{B}}
 \end{array}$$

Unary **W** seems empirically undesirable. Szabolcsi (1989) observes that we have yet to find a language in which an expression related to the one below means *John turns himself*. It would require unary **W** as shown.

$$\begin{array}{c}
 (29) \textit{John turns} \\
 \hline
 \text{NP} \quad (\text{S}\backslash\text{NP})/\text{NP} \\
 \hline
 \text{S}\backslash\text{NP}^{\text{W}} \\
 \hline
 \text{S}^{\text{app}}
 \end{array}$$

The point of course is not that the word ‘turn’ *might* mean ‘turn himself’ in this example, but in a syntacticized system where combinators do their work by syntactic types, i.e. by being opaque to the lexical meaning of *turn*, the rule above would also engender *John reads*, *John devours*, to mean John reads himself and John devours himself. Similarly, a binary **W** is problematic. The same example can be derived by binary **W** as follows:

$$\begin{array}{c}
 (30) \textit{John turns} \\
 \hline
 \text{NP} \quad (\text{S}\backslash\text{NP})/\text{NP} \\
 \hline
 \text{S}^{\text{W}}
 \end{array}$$

Thus we have good empirical reasons not to have **W** in syntax at all. This result might appear to make the ternary **S** nonredundant (see 27). First I note from Szabolcsi (1989) that binary **S** is certainly operating in syntax because we know the existence of languages with parasitic gaps. The crucial involvement of **S** is shown below.

$$\begin{array}{c}
 (31) \textit{(articles) which I will file without reading} \\
 \hline
 \text{VP/NP} \quad (\text{VP}\backslash\text{VP})/\text{C}_{\text{ing}} \quad \text{C}_{\text{ing}}/\text{NP} \\
 \hline
 (\text{VP}\backslash\text{VP})/\text{NP}^{\text{B}} \\
 \hline
 \text{VP/NP}^{\text{S}}
 \end{array}$$

Steedman (1988)

It is **S** semantics because *articles* is an argument of both *file* and *read*, and without the first “gap” after *file*, it is ungrammatical, say **articles which I will file the folders without reading*.⁵⁰

Further evidence is from coordination: *the articles which I will file without reading and report without contradicting*. Now the redundancy of ternary **S** follows from the necessity of binary **S** and application (likewise the redundancy of ternary **B**, which also follows from binary **B** and application):

$$(32) \quad \frac{\frac{\omega_1 \quad \omega_2 \quad \omega_3}{(X/Y)/Z: f \quad Y/Z: g \quad Z: a}}{X/Z: \mathbf{S}fg}}{X: \mathbf{S}fga = fa(ga)}^{\text{app}}$$

What about unary **B** and unary **S** operating in syntax? Recall some syntacticized versions of these combinators in order to study what is at stake.

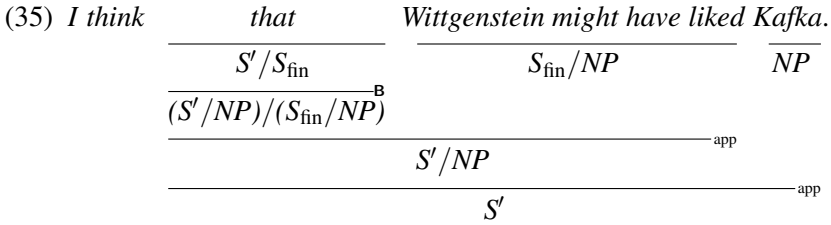
$$(33) \quad \begin{array}{ll} \text{a. } X/Y \leftrightarrow (X/Z)/(Y/Z) & ({}_1\mathbf{B}) \\ \text{b. } X/Y \leftrightarrow (X \setminus Z)/(Y \setminus Z) & (>_1\mathbf{B}_\times) \\ \text{c. } (X/Y)/Z \leftrightarrow (X/Z)/(Y/Z) & ({}_1\mathbf{S}) \end{array}$$

A revealing empirical argument against unary **B** came from Szabolcsi (1989), who suggested that the syntactic behavior of complete constituents does not necessarily extend to incomplete constituents, which is precisely the effect of unary **B**.

Consider the complementizer *that*, with the category S'/S_{fin} . We would not want the incomplete version $(S' \setminus NP)/(S_{\text{fin}} \setminus NP)$ which would be engendered by unary **B**:

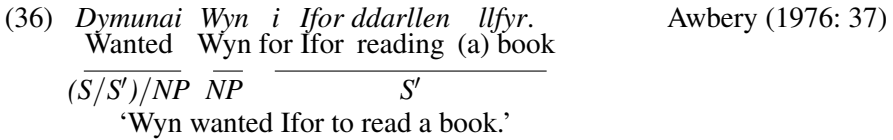
$$(34) \quad \begin{array}{l} \text{a. } I \text{ think } \textit{that} \textit{ Wittgenstein might have liked Kafka.} \\ \frac{\frac{VP/S' \quad S'/S_{\text{fin}} \quad S_{\text{fin}}}{S'}}{\textit{that} \quad \textit{might have liked Kafka}}^{\text{app}} \\ \text{b. } *I \text{ think } \textit{Wittgenstein} \quad \textit{that} \quad \textit{might have liked Kafka} \\ \frac{\frac{S'/S_{\text{fin}} \quad S_{\text{fin}} \setminus NP}{(S' \setminus NP)/(S_{\text{fin}} \setminus NP)}^{\mathbf{B}}}{S' \setminus NP}^{\text{app}} \end{array}$$

Some complementizers in some languages might choose to make their version of unary **B** grammatical, but this would have to be a lexical choice, not engendered by syntax. In fact, English does just that: the forward variety of unary **B**, viz. $(S' \setminus NP)/(S_{\text{fin}} \setminus NP)$ gives exactly the same semantics as (34a):

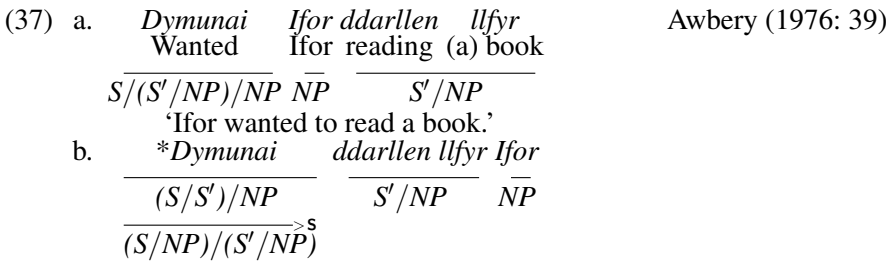


There must be language-specific constraints on unary **B**, for example divide by ‘/NP’ rather than ‘\NP’ as above, hence it must be lexicalized.⁵¹

Now consider Welsh to see the effects of a freely-operating unary **S**. Welsh has a strict word order of VSO, which can be characterized as VSS’ when the argument is a complement clause S’. We can categorize the complement-taking verb as such:



A unary **S** must be lexically constrained because, although Welsh allows subject-sharing complements (37a) (i.e. incomplete constituents), the word order instigated by unary **S** from complement-taking verbs would be ungrammatical (37b).⁵²



The Welsh verb must avoid unary **S**. The modalities cannot help in such examples to eliminate them. Therefore unary **S** cannot be syntactically free.

Let us now take stock of what is needed in syntax in terms of dependency and constituency, and what should be lexically controlled. The combinators **S**, **K**, **C**, **B**, **W** and **I** play a crucial role in establishing the power of combinators to capture any computable semantic dependency. The first two are Schönfinkel’s primitives, and the last four were Curry’s primitives until he encountered Schönfinkel’s work in a literature search in 1927. He adopted **K** immediately, and considered **S** to be somewhat artificial.

William Craig proves in his section §5H of Curry and Feys (1958) that among this group an **S**-effect is impossible without **B**, **C** or **W**. Therefore **K** can be ignored for the **S**-effect. A **K**-effect is impossible with the remainder of the group. Without **S** and **K**, a **B**-effect is impossible. A **W**-effect is possible without **K**. Thus $\{\mathbf{I}, \mathbf{K}\}$ and $\{\mathbf{B}, \mathbf{S}, \mathbf{C}, \mathbf{W}\}$ form two sets in which any system that aims at behavioral equivalence to lambda calculus must contain one combinator from each set.

S and **B** are not interdefinable if we eliminate **C** and **K**. Similarly, **C** and **W** are not interdefinable if we eliminate **S** and **B**. On what basis do we choose a set of combinators that always operates on syntax? Szabolcsi offers a formal criterion in addition to the empirical ones we have seen so far:

- (38) The combinators running free in syntax are (a) noninterdefinable, or (b) compositions of such noninterdefinable combinators. Other derived combinators are lexicalized. Szabolcsi (1989: 305)

This hypothesis is not sufficient to rule out **K** and **I** from syntax. **K** is not interdefinable by the remaining five combinators, and without **K**, **I** is not interdefinable by **BSC** either. The criterion therefore is meant to supplement the empirical reasons rather than replace them.

The following desiderata emerge from interdefinability and from the limits of dependencies attested in natural languages:

- (39) (i) **K** is not desirable because (a) its lexical effect has not been attested in languages, (b) its power of deletion is a threat to decidability.
 (ii) Any slash is implicitly an **I** in terms of semantic dependency. **I** adds nothing to syntax, but it must play a crucial role in the lexicon.
 (iii) **B** seems inescapable, otherwise we cannot surpass the context-freeness barrier. Application is good enough for context-free dependencies (Bar-Hillel, Gaifman and Shamir 1960). Without **K**, **B** cannot be defined by **SCIW**. With **IK** gone from syntax, the remainder **SCW** cannot achieve a **B**-effect.
 (iv) Some manifestation of the **CW** effect is needed, which brings **S** into the discussion. This can be done by the sequence **BST** because $\mathbf{C} = \mathbf{B}(\mathbf{T}(\mathbf{BBT}))(\mathbf{BBT})$ as Church (1940) and Szabolcsi (1989) noted, and $\mathbf{W} = \mathbf{ST}$. It can also be done by **BCW** because $\mathbf{S} = \mathbf{B}(\mathbf{B}(\mathbf{BW})\mathbf{C})(\mathbf{BB})$.

Cases (i) and (iv) need empirical support. No language seems to have the **K**-like vacuous abstraction exemplified below:

(40) * *WHAT does Mary like Bill?* Szabolcsi (1989: 3b)

Notice that this is different than the apparently related German example below, where *wh-in situ* is grammatical. There is no vacuous abstraction here, since *wers* are one and the same.

(41) *Wer glaubst du wer nach hause geht?* Crain and Pietroski (2001)
Who do you think who goes home?

The closest example I could think of for vacuous abstraction is the headed morphological compounds of German (42): “Genitive case endings function as morphological “glue” when their use would be disallowed in the corresponding noun phrase” Payne (1997: 93).

(42) *Bischoff-s-konferenz* (Anderson 1985)
bishop-GEN.sg-conference
‘conference of bishops’
* for ‘conference of bishop’

The process is quite productive, and from the perspective of the constituents of the compound, *-s-* seems like **K**’s victim, with the semantics $\mathbf{K}(b'k')gen' = b'k'$. The primed semantic objects stand for the semantics of *Bischoff*, *Konferenz* and *-s-* respectively.

But it could also be that *-s-* is another lexical item in German, different than its genitive case marker interpretation, which yields a morphologically-headed compound as Payne suggested. Thus if we are willing to extend our notion of lexicon to include objects with categories other than words, we have an analysis without **K**. No such freedom seems to exist for *what* in (40).⁵³

As for the **CW** effects, we have seen empirical reasons for **W** not to operate in syntax, therefore it must be lexicalized. The question then is the following: do we lexicalize **C**, or is it free in syntax in some arity? According to Szabolcsi’s formal criterion (38), its lexicalization depends on whether we have **T** in syntax, because $\mathbf{C} = \mathbf{B}(\mathbf{T}(\mathbf{BBT}))(\mathbf{BBT})$.

A freely-operating **T**, in the truest sense of the term, is redundant in binary form if the unary version is available (§4.1). And, as we shall see, the unary version must be available in syntax in a constrained way. These results altogether suggest that **C** must be lexical.

Empirical reasons complement the picture by suggesting lexicalization as well. Take for example the VSO language Welsh. The category of the transitive verb is $(S/NP_2)/NP_1$, where NP_1 stands for the subject NP for convenience (Welsh has no morphological case). Unary **C** would yield

$(S/NP_1)/NP_2$, which is equivalent to saying that VOS order would be grammatical too, which is not true for Welsh. A binary **C** would yield S/NP_1 from the configuration $(S/NP_2)/NP_1 NP_2$, which also amounts to licensing VOS for Welsh. Judging from Steele's (1978) typological study of limited appearance of alternative word orders, this process must be lexically controlled in all languages.

The occurrence of strict word-order languages suggest that we should not employ **C** to understand the free word-order effects of scrambling languages, unless we are willing to entertain parametric competence grammars where one set of combinators prevails over others depending on some kind of parameter setting over the universal repertoire. As there is no initial-state universal grammar in CCG that "grows into" an adult-state grammar, there is no room for a parametric combinatory base either, thus the prediction of CCG is that any **C**-effect must be specified in the lexicalized grammar of a language.

Next I show that the syntactic common core of CCG, the **BTS** system, is computationally well supported. Then we look at the additional assumptions about combining variable-free syntax with variable-friendly semantics in the next chapter.

5. The **BTS** system

Adjacency as an auxiliary assumption was deemed detrimental because combinators cannot handle wrap (§1). A **C** in the lexicon is not wrap because it does not wrap strings but syntactic and semantic types. Recall Szabolcsi's (1989) category $(S \setminus NP)/NP/PP$ for *introduce*, rather than the surface word order $(S \setminus NP)/PP/NP$, which was motivated by binding possibilities, which required unary **C** to apply lexically to $(VP/NP) \setminus (VP/PP/NP)$. We may consider this move as the abandonment of some nonconstituent coordination analysis (43), but this is an issue within reach of combinators, and its resolution is not our concern here. It is important that it does not violate adjacency.

(43) *John announced Mary and introduced Harry to the party crowd.*

We take adjacency as a fundamental assumption to look at its full consequences, rather than bring it in when necessary.

The mild context-sensitivity result of Vijay-Shanker and Weir (1994) for CCG holds only if a bounded use of powers is employed, i.e. \mathbf{B}^n and \mathbf{S}^m for some m, n . Recall that $\mathbf{B}^n = \mathbf{BBB}^{n-1}$ and $\mathbf{S}^m = \mathbf{BS}^n \mathbf{S}^{m-1}$. The second

clause of (38) predicts their free operation in syntax because it equivalent to **BXY** for some noninterdefinable *X* and *Y*, as Szabolcsi observed.

Hoffman (1993) showed that a freely-operating **T** gives us the strictly nonlinear-indexed language $\{a^n b^n c^n d^n e^n \mid n \geq 0\}$. Current findings on the adequacy of nearly context-free grammars and the inadequacy of context-free grammars for linguistic description depend on the bounded use of **B** and **T**.

The **T** must be finitely schematized to maintain near context-freeness, which can be done by compiling over a radically lexicalized grammar to see all kinds of argument and result types. Alternatively, it can always be kept in the lexicon, which by definition would be a finite schematization. Let us consider both possibilities.

The lexical **T** is by definition a unary **T**. Recall also that binary **T** is rendered redundant by the unary **T** and the primitive of the system.

Regarding the possibility of a unary **T**-less syntax, it is not possible to always build **T** into the lexical categories of argument encoders such as determiners and case markers. Some languages lack determiners. Moreover, there are caseless languages, and also languages with morphological case where we need **T** in syntax although case is not involved. Consider some Turkish data in this regard.

- (44) [*Gelin-e ben-im uyu-duğ-um-u*], [*damad-a Ahmet'in çalış-tığ-ı-nı*]
 Bride-DAT I-AGR.1s sleep-COMP-1s-ACC groom-DAT A-AGR.3s work-COMP-3s-ACC
söyle-miş.
 tell-PERF
 lit. 'S/he told the bride that I am sleeping and the groom that Ahmet is working.'

The string *ben-im uyu-duğ-um-u* must be type-raised (by **T**) and composed (by **B**) with *gelin-e*, so that we can account for the unorthodox constituency of *Gelin-e ben-im uyu-duğ-um-u* in coordination. This is shown below. The second coordinand must do the same for its constituents.

- (45)
- | | | | |
|---|---------------------------------|----------------------------|---|
| <i>gelin-e</i>
bride-DAT | <i>ben-im uyu</i>
I-1s sleep | <i>-duğum</i>
-COMP.1s | <i>-u</i>
-ACC |
| NP_{dat} | S_{1s} | $S'_{1s} \setminus S_{1s}$ | $(S \setminus NP_{nom} \setminus NP_{dat}) / (S \setminus NP_{nom} \setminus NP_{dat} \setminus NP_{acc}) \setminus S'$ |
| | | | $(S \setminus NP_{nom} \setminus NP_{dat}) / (S \setminus NP_{nom} \setminus NP_{dat} \setminus NP_{acc})$ |
| $\xrightarrow{\text{T}}$ | | | |
| $(S \setminus NP_{nom}) / (S \setminus NP_{nom} \setminus NP_{dat})$ | | | |
| | | | $\xrightarrow{\text{B}}$ |
| $(S \setminus NP_{nom}) / (S \setminus NP_{nom} \setminus NP_{dat} \setminus NP_{acc})$ | | | |

Leaving **T** to the lexical category of a case-marker such as the accusative case on the nominalized verb, as is done above for *uyu*, will not always work, because unmarked clauses must be type-raised as well in certain syntactic contexts:

- (46) *Gelin-ce ben-in uyu-duğ-um, damad-ça da Ahmet'in çalış-tığ-ı*
 Bride-ESS I-AGR.1s sleep-COMP-1s groom-ESS A-AGR.3s work-COMP-3s
bil-in-iyor.
 know-PASS-PROG
 lit. 'It is known by the bride that I am sleeping and by the groom that Ahmet is working.'

Unless we lexicalize all Turkish subordinate clauses, which can be case-marked or unmarked nominalized clauses, **T** must be a lexical rule.⁵⁴

Another empirical reason for a schematized **T** is the word-internal recursion in nominals. The Turkish relativizer suffix *-ki* can be attached to case-marked nouns whose case relation is one of possession, time, or place (i.e., the genitive and the locative), for example *ev-in-ki* (house-GEN-ki 'the one of the house') and *ev-de-ki* (house-LOC-ki 'the one in the house').

Its effect is to create a nominal stem on which all inflections can start again. As Hankamer (1989) noted, there is no upper bound on this process of relativization (e.g. *ev-i-nde-ki-ler-in-ki-ler-de-ki*).

It follows that these words must be derived in syntax (otherwise we would have an infinite lexicon). They can take part in nontraditional constituencies such as those below, which is possible in CCG only if these words are type-raised and composed, therefore type raising must be a rule. The critical step is shown in (47b).

- (47) a. [*Ev-de-ki-nin-ki adam-a*], [*salon-da-ki çocuğ-a*] *sarılmış*
 house-LOC-ki-GEN-ki man-DAT room-LOC-ki child-DAT hug-PERF
 lit. 'The one in the house's one hugged the man, and the one in the room the child.'
 e.g. 'The friend/acquaintance of the one in the house hugged the man, and the one in the room the child.'

- b.
$$\frac{\frac{NP}{S/(S\backslash NP)} \quad \frac{(S\backslash NP)/(S\backslash NP\backslash NP_{\text{dat}})}{\quad}}{S/(S\backslash NP)} \quad \text{B}$$

Thus the only theoretical possibility to maintain near context-freeness of CCG and to have a **BTS** system, given our current understanding, is to finitely schematize the unary **T** as a universal lexical rule. Every language has

a finite vocabulary of argument categories, therefore it seems to be a feasible solution. Since by this choice we keep **B** and **T** in syntax, **C** can be lexical. Because we do not keep **W** or **C** in syntax, **S** can be syntactic.

We can now have a look at variable-friendly semantics in relation to **BTS** syntax.

Chapter 6

The LF debate

This chapter is about apparently the least adjacency-related and most post-PADS related aspect of combinatory theorizing: the issue of having a Logical Form (LF) for narrowing down the possible interpretations, without a concomitant narrowing of possible constituents.⁵⁵ The issue is also the most divisive and perplexing.

The reader is referred to better summaries and historical accounts such as Szabolcsi (1989, 1992, 2003), Jacobson (1999, 2002), Barker and Jacobson (2007), Steedman (1996a, 2011). I will reiterate their way of handling some referential-interpretive phenomena, along with some assessment and predictions.

Empirical concerns about constituency force the CCG variants to converge on a **BTS** syntax, where **T** must be constrained by the lexicon, either by type raising all the argument types in the lexicon, or by operating the unary rule under a limited domain and range, which can be compiled from the lexicon. Adding unary **BCWZ** to this base where **B**, **C** and **W** are constrained by the lexicon (for example apply unary **B** to objects only, unary **C** to two- or more-complement verbs, unary **W** to reflexives, and unary **Z**, viz. **BSC**, to pronouns), is where CCG models begin to differ.

The **BTS** system alone is variable-free syntax that makes use of bound variables in epitheorems only (in Curry's sense; see the discussion in page 31), related to the predicate-argument dependency structures (PADS). These are the systems with a logical form, i.e. they employ a lexical use of unknowns rather than variables. **BCWZ** systems on the other hand amounts to variable-free semantics, in addition to variable-free syntax. Binding of anaphors is handled by combinators as well, such as Jacobson's **Z** and a special unary **B**, and Szabolcsi's **W** in the lexicon, which eschews Bach-style wrap, which has no combinatory counterpart.

Jacobson (1999), Steedman (1996a, 2000b, 2011), Szabolcsi (1989) summarize what is at stake for each path. Szabolcsi's and Jacobson's arguments are both methodological, to culminate variable-free syntax with variable-free semantics, and empirical, for example whether we distinguish *John left* and *He left* syntactically, the first one as a sentence whose denotation is a proposition, and the second as a function from an individual to a proposition. Steed-

man's argument is from automata-theoretic concerns, to reduce the amount of nondeterminism engendered by unary rules and eliminating additional resource management needs such as a quantifier store, and also from cognitive science. He contrasts syntax-specific command relations which seem to defy traditional concepts such as c-command (e.g. an argument can be relativized independent of its c-commanding position) with the bound-element behavior, which seems to faithfully maintain such relations (e.g. reflexives and reciprocals), suggesting a branching evolutionary pathway at work. Recall Steedman's argument that reference avoids combinators and depends on logical form, which he suggests might arise out of pressures for speedy processing.⁵⁶

There is another perspective that seems to call for a closer look at the problem of LF. The syntactic dependencies engendered by syntactic processes are strict about the crossing or nesting kind (1a–b). But the semantic dependencies manifested by quantifiers and pronouns can cross *and* nest (1c–d).

- (1) a. *A violin_i which this sonata_j is easy to play_j on_i*
- b. **A sonata_i which this violin_j is easy to play_i on_j*
- c. *Every man_i thinks that every boy_j said that his_j mother loves his_i dog.* (Jacobson 1999)
- d. *Every man_i thinks that every boy_j said that his_i mother loves his_j dog.*

The lexical predicate-argument structure and the semantic dependencies it represents, the PADS, must be distinguished from the notion of LF. The linguistic notion of LF is borrowed from logic, where it meant, through the works of Frege, Carnap, Russell, early Wittgenstein, Tarski, culminating in Montague (1974), a pristine form of logical aspects of a sentence cleared off the surface characteristics such as inflection, agreement, word order, etc.

Chomsky's (1976) and May's (1977, 1985) LF is a structural domain at which not-so-pristine issues such as quantifier movement and semantic reanalysis are handled, to the extent of having a separate syntax such as in Pesetsky (1985, 1995). In logician's case, nothing intervenes to provide a model theory for LF (except some model-stage semantic storage and reinterpretive operations) because scope and predicate locations are all in place, whereas in transformational linguist's case conditions must be predicated over LF to get them, and more significantly, we need covert operations of different kinds to get the right LF. The closest analogue of such operations in Montague is the quantifying-in rule, which introduces a prosodic variable to be substituted by a logical formula.

In this sense Chomsky's (1981) binding conditions A, B, C in (2) can be looked at from two angles: (a) As theory-internal constraints at some level of representation, such as LF as an interface, or, as in earlier transformational accounts, as a constraint on the input and output of transformations, (b) as desiderata for any theory to account for the syntactic narrowing of reference. They are roughly reformulated below to avoid theory-specific terminology:

(2) Condition A: An anaphor (reflexive or reciprocal) must be bound in a minimal tensed domain.

Condition B: A pronoun must be free where an anaphor must be bound.

Condition C: A referring expression must be free everywhere.

We have seen options (a–b) implemented in CCG various ways: (i) the adoption of LF as a level, without a model-stage extra storage or reinterpretation, with conditions such as LF-command but without any special syntax associated with it. This is Steedman's (2011) *surface compositionality*, which means every surface constituent is interpretable, with any unresolved reference in it bound either by tandem deterministic LF operations in the course of a derivation, or left to discourse. (ii) The LF-less narrowing of syntactic types in the lexicon by a lexical use of unary combinators (Szabolcsi 1992). (iii) The traditional Montagovian LF-less model with unary rules and lexical types for initiating, projecting and binding of bound pronominals, leading to Jacobson's (1999) *direct compositionality* ("direct" in the sense that every semantic object that is compositionally derived is model-ready).



As the brief descriptions suggest, the proposals conceive different ways to narrow down possible categories. Let us look at each alternative in some detail.

1. Steedman's LF

Steedman (1996b) defines *LF-command* as a substantive constraint on possible categories, which is predicated over the LF. It is in this sense that LF is the only structural level of representation in Steedman's CCG, all other constraints for example on syntactic types and derivational structures are completely eliminated by radical lexicalization. I provide a newer formulation of LF-command from Steedman and Baldrige (2011).

- (3) A node α in a logical form Λ LF-commands a node β in Λ if the node immediately dominating α dominates β and α does not dominate β . (LF-command)

The LF unknowns are of the kind *and'* x , *pro'* x , which are nonbranching pro-terms where x is identical to some element in the LF. In other words, *and'kinski'kinski'* is (4a) rather than (4b).

- (4) a.  b. 

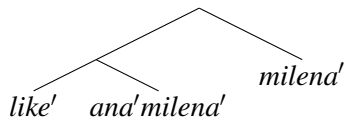
His binding theory reduces to one condition, which is similar to Condition C.

- (5) No node except the argument in a pro-term can be LF-commanded by itself. Steedman and Baldridge (2011) (Condition C)

This condition eliminates (6a–b) as possible interpretations of otherwise grammatical examples. Condition A and Condition B are explained away by noting that reflexivization is lexicalized (i.e. it requires the lexical category of a verb), and pronominal binding (of x in *pro'* x) is not lexicalized.

- (6) a. *She*_{*i} *liked Milena*_i.
 b. *I*_{*j} *think she*_{*i} *liked Milena*_{i|j}.
 c. *Milena*_i *liked her*_{*i}/*herself*.

Thus *herself* in an example such as (6c) would have access to all the arguments in the LF of $\lambda x_1 \lambda x_2. \textit{like}'x_1x_2$, which means it can only substitute for x_1 . If *herself* has the semantics $\lambda P \lambda x. P(\textit{and}'x)x$, then we get LFs of the sort in (7) once it combines with the verb.

- (7) 

The analysis of *her* in (6c) is the main source of variation in Steedman's CCG. Although there seems to be a recent consensus that condition B effects should be left to a discourse model (Jacobson 2007, Steedman 2011), there is some work done in LF in Steedman's case to eliminate proliferating readings in examples such as below. He avoids semantically powerful yet syntactically innocuous operations such as an extra stack for scope-taking or the semantics-only type-change, which could in principle dispense with LF for handling this kind of work.

- (8) a. *Every farmer who owns a donkey_i feeds it_i.*
 b. *All the girls admired, but most boys detested, one of the saxophonists.*
 Geach (1972)

Steedman's (2011) suggestion is that, unlike the deletion accounts of transformationalism, which deliver too many readings for examples like (8b), and unlike strict Montagovianism, which would require extra devices on the semantic side for (8a–b), assuming an LF may give us surface-compositional readings only, with concomitant syntactic assumptions such as the type-raising of all arguments but generalized quantification of only the universal quantifiers.

This is where his LF assumption begins to do more work than reflexivization and nonsubject pronominal binding. His Skolem terms, which are LF terms in need of a scoping universal quantifier, gets the scope information and the terms of skolemization from LF-command.

Although Steedman's introduction of Skolem terms in place of nonuniversal NPs gives us only the possible readings in (8), example (9a) is susceptible to his LF-term binding although there is no Skolem term, hence we need Condition B effects to rule it out. And, (9b)'s Skolem-term is not sufficient to eliminate binding in LF to *it*. We need to call in yet again condition B effects of discourse to the rescue.

- (9) a. *Every donkey_i feeds it_{*i}.*
 b. *A donkey_i feeds it_{*i}.*

Thus Skolem terms and their tight management during the syntactic process sometimes need discourse conditions anyway, to find their antecedents. This is true of "donkey anaphora" as well. Consider (8a) in a context where a donkey named *Balthazar* is left to the common goodwill of the village, which gives us a free interpretation of *it*.

We have yet to find cases where a quantifier-bindable pronoun can only have that reading. That would vindicate an exclusively grammatical solution to pronoun resolution in at least some constructions. We also have examples like (10), where an antecedent *within* a quantified NP not c-commanding (or LF-commanding) the pronoun is possible.

- (10) *Every professor's_i neighbor respects her_i.* (Postal and Ross 2009:ex.66)

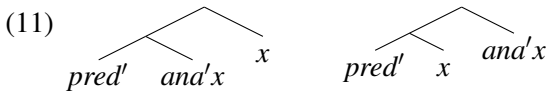
If this were the only reading, it would jeopardize a Skolem-binding solution of bound anaphora over an LF structure, because the potential antecedents

of quantifier-bound pronouns are read off in the theory as the list of LF-commanding terms.

As it currently stands, Steedman's LF-Skolem-command account must leave both bound and free interpretations of (10) to discourse.⁵⁷

2. Szabolcsi's reflexives

One useful consequence of assuming an LF-command and pro-terms is that we can universally rule out subject reflexives such as **sheself* without an appeal to **W** or **Z**, thus without having to stipulate this constraint in every lexicalized grammar. The LF $pred'(and'x)x$ satisfies Condition C, but $pred'x(and'x)$, which would be engendered by the LF of **sheself*, does not:



Szabolcsi's (1992) combinatory solution below to the same problem is LF-less therefore without c-command or its LF equivalent. Her claim is that the binding theory of (2) follows from combinatory assumptions about syntax-semantics, including the lexical assumptions about the predicate-argument structures. The relevant combinatory options are the lexical use of **W** and **B**.

- (12) a. $sheself := *S/(S\NP_{3s})$
 b. $herself := (S\NP)/((S\NP)/NP): \lambda f \lambda x.fxx$

Example (12a) is an illicit type because the explicit involvement of **W** for reflexives presumes that we have a function with two or more arguments in the predicate-argument structure to begin with, which is inconsistent with this syntactic type. Assuming that subjects are universally type-raised, like all arguments, the impossibility follows without further conditions.⁵⁸ That explains (13a) but not (13b), as Szabolcsi pointed out.

- (13) a. *Sheself left.* Szabolcsi (1992)
 b. **Sheself sees everyone.*

The second example would require the category $(S/NP)/((S\NP)/NP)$ for the nonsubject argument if it were grammatical. This would be different than (12b), as expected, but (12a) would allow it if we let unary **B** loose in syntax (divide 12a by $'/NP'$), which is eliminated for independent reasons. This takes care of Condition A without an LF, as a consequence of the syntax and semantics of **B** and **W**.

A further condition is imposed on the lexicon: reflexives must apply to lexical items only, otherwise (14a) would be allowed. Lexicalization is needed because (14b) must be derivable, which shows that there are syntactically derived $(S \setminus NP)/NP$ types.

- (14) a. **Mary believes that John loves herself.* Szabolcsi (1992)
 b. *Who does Mary believe that John loves?*

$$\frac{\frac{(S \setminus NP)/S' \quad S'/S \quad S/(S \setminus NP_{3s}) \quad (S_{fin} \setminus NP_{3s})/NP}{(S \setminus NP)/S} \xrightarrow{B} \quad \frac{S/NP}{S/NP} \xrightarrow{B}}{(S \setminus NP)/NP} \xrightarrow{B}$$

I write the lexical constraint (the +LEX feature of the slash in Steedman and Baldrige 2011), as ‘ ℓ ’ or ‘ $\setminus \ell$ ’, with the interpretation that an item e.g. $\alpha := A \setminus \ell B$ requires a leftward type B to be lexical to yield A (likewise ‘ ℓ ’ for the rightward variety):

- (15) B must be the type of a lexical item (the LEX convention)
 in $A \setminus \ell B$ and $A \ell B$.

Now the string *believes that John loves* bear the -LEX value, which accounts for (14) because *herself* bears the ‘ $\setminus \ell$ ’ (+LEX) constraint.

With or without LF, some right-node raising examples are forced to an ellipsis analysis under the lexicalization of reflexives. The coordinate structure below does not bear a lexical type.

- (16) *Kinski adored and Wittgenstein hated himself.*

The LF proposal is forced to a semantic “wrap” (i.e. **C**) analysis in English ditransitives, and for VSO languages. I repeat Steedman and Baldrige’s treatment of reflexives below to elaborate.

- (17) *Mary saw herself.*

$$\frac{\frac{S/(S \setminus NP_{3s}) \quad (S \setminus NP)/NP \quad (S \setminus NP_{3s}) \setminus \ell ((S \setminus NP_{3s})/NP)}{\lambda x \lambda y. see'xy} \quad \lambda P \lambda y. P(ana'y)y}{S \setminus NP_{3s}: \lambda y. see'(ana'y)y} \leftarrow$$

The innermost lambda abstraction of three or more arguments is unavailable to the reflexive with its $\lambda P \lambda y. P(ana'y)y$ semantics. We must schematize the types of *herself* to get the right semantics for these cases, which is nontrivial because it involves semantic wrap to get x in between *ana'y* and y below. This is harmless computationally because it is done in the lexicon.

$$\begin{array}{c}
 (18) \text{ Mary} \quad \text{gave} \quad \text{herself} \quad \text{a present.} \\
 \hline
 \frac{(S \setminus NP) / NP / NP \quad ((S \setminus NP_{3s}) / NP) \setminus_{\ell} ((S \setminus NP_{3s}) / NP / NP)}{\lambda x \lambda y \lambda z. \text{give}'xyz \quad : \lambda P \lambda x \lambda y. P(\text{and}'y)xy} \\
 \hline
 (S \setminus NP_{3s}) / NP: \lambda x \lambda y. \text{give}'(\text{and}'y)xy <
 \end{array}$$

We are similarly forced to an analysis involving semantic wrap in VSO languages (19a). (For brevity, NP^{\uparrow} represents a type-raised NP.) First, notice that the +LEX constraint applies to Welsh reflexives as well, although they are not string-adjacent to the verb like in English. Note also the knowledge of LF, where $x(\text{and}'x)$ rather than $(\text{and}'x)x$ is assumed for Welsh, because of VSO verbs, and also because of (19b).⁵⁹

$$\begin{array}{c}
 (19) \text{ a.} \quad \text{Gwelodd} \quad \text{Wyn} \quad \text{ef ei hun} \\
 \text{Saw} \quad \text{Wyn} \quad \text{himself} \\
 \hline
 \frac{S / NP / NP_{3s} \quad (S / NP) \setminus (S / NP / NP_{3s}) \quad S \setminus_{\ell} (S / NP / NP_{3s}) \setminus NP^{\uparrow}}{\lambda x_1 \lambda x_2. \text{see}'x_2x_1 \quad : \lambda f. fw' \quad : \lambda P \lambda Q. P(\lambda x. Qx(\text{and}'x))} \\
 \hline
 S \setminus_{\ell} (S / NP / NP_{3s}): \lambda Q. Qw'(\text{and}'w') < \\
 \hline
 S: \text{see}'(\text{and}'w')w' <
 \end{array}$$

‘Wyn saw himself.’

Awbery (1976: 131)

- b. **Gwelodd ef ei hun Wyn*
Saw himself Wyn

The LF-less **W** semantics and lexical syntactic types for reflexives generalize nicely to $\lambda P \lambda Q. P(\lambda x. Qxx)$, as shown below, as an alternative to the LF account in (19a).

$$\begin{array}{c}
 (20) \quad \text{Gwelodd} \quad \text{Wyn} \quad \text{ef ei hun} \\
 \text{Saw} \quad \text{Wyn} \quad \text{himself} \\
 \hline
 \frac{S / NP / NP_{3s} \quad (S / NP) \setminus (S / NP / NP_{3s}) \quad S \setminus_{\ell} (S / NP / NP_{3s}) \setminus NP^{\uparrow}}{\lambda x_1 \lambda x_2. \text{see}'x_2x_1 \quad : \lambda f. fw' \quad : \lambda P \lambda Q. P(\lambda x. Qxx)} \\
 \hline
 S \setminus_{\ell} (S / NP / NP_{3s}): \lambda Q. Qw'w' < \\
 \hline
 S: \text{see}'w'w' <
 \end{array}$$

3. Jacobson’s pronouns

Jacobson’s starting point is that syntactic elements that seem like variables, for example pronouns, do not necessitate variables in syntax or semantics. Working with combinatory-syntactic assumptions, she avoids transformational-style variables from the beginning (the empty categories),

and suggests a binding scenario which takes place within the semantics of a specialized unary **Z** (specialized to apply to *e*-type NPs only, hence properly equipped to bind the right kind of pronouns-as-variables). With the help of a specialized unary **B** called *g* (for 'Geach'), this move avoids the use of LF to account for the bound and free interpretations of pronouns. This way pronouns-as-arguments are forced to yield functions rather than propositions, therefore they make a finer distinction in possible syntactic types and bear empirical consequences.

Her narrowing of the possibilities in the grammar-lexicon are roughly as follows. The reader is referred to Jacobson (1999) for full exposure, and to Barker and Pryor (2010) for a computational model using monads (i.e. threading of *g*-computations with *z*-computations).

Pronouns are lexically (*e, e*)-types in her theory, which she translates syntactically as $NP.NP$. Syntactically this is the collection of all functions from NP types to NP types. I will call them *exponent types* for easier reference. It is conceived as a semantic narrowing of an NP with syntactic significance, because of the distinction from another collection of functions from NPs to NPs: $NP|NP$.

The exponent types must be mediated in syntax to force an individual-to-proposition functional readings of (21a–b), rather than the propositional ones in (21c–d), because the verbs lexically do not know the distinction. This is a compelling argument for the syntactic narrowing of type *S*.

- (21) a. *He left.* (S^{NP})
 b. *Kafka adored her.* (S^{NP})
 c. *John left.* (*S*)
 d. *Kafka adored Milena.* (*S*)

This Jacobson achieves with a specialized unary **B**, where $Z=NP$; cf. the syntactically freer one in §4(28).

$$(22) X|Y: f \rightarrow X^Z|Y^Z : \lambda g \lambda x. f(gx) \quad (g-Z)$$

Since this is not syntactic **B**, the slash can bear any modality, not just ' \backslash_{\diamond} ' or ' \backslash_{\circ} '. We shall see later that this is further corroborated by the data; (38b) needs to apply this rule when the slash is ' \backslash_{\star} '.

Now we can derive (21b) as a function from individuals to propositions, syntactically $S.NP$. This is different than deriving it as $S|NP$ with the freer version of unary **B** because, syntactically speaking, the expression needs no arguments.

$$\begin{array}{c}
 (23) \quad \frac{\frac{Kafka}{S/(S\backslash NP_{3s})} \quad \frac{adored}{(S\backslash NP)/NP} \quad \frac{her.}{NP^{NP}}}{\frac{(S\backslash NP)^{NP}/NP^{NP}}{(S\backslash NP)^{NP}} \rightarrow} \\
 \frac{\frac{S^{NP}/(S\backslash NP_{3s})^{NP}}{S^{NP}} \rightarrow}{}
 \end{array}$$

Jacobson’s way of handling the pronouns therefore needs no lexical distinction between a contextually bound but syntactically free use of a pronoun, and a syntactically-bound pronoun. They both derive functions rather than propositions. I show the semantics to make this point explicit. Notice that the variable z below is not a syntactic argument because the syntactic type is not $S|NP$.

$$\begin{array}{c}
 (24) \quad \frac{\frac{He}{NP^{NP} : \lambda x.x} \quad \frac{left.}{S\backslash NP : \lambda y.leave'y}}{\frac{S^{NP}\backslash NP^{NP} : \lambda f \lambda z.leave'(fz)}{S^{NP} : \lambda z.leave'z} <}
 \end{array}$$

The bound pronoun below is where her unary **Z** does its binding. This combinator is specialized in Jacobson’s case to apply to NPs only; cf. the freer version §4(54).

$$(25) \quad (X|_i NP)|_j Y : f \rightarrow (X|_i NP)|_j Y^{NP} : \lambda g \lambda x.f(gx)x \quad (z-NP)$$

$$\begin{array}{c}
 (26) \quad \frac{\frac{John}{S/(S\backslash NP_{3s})} \quad \frac{loves}{(S\backslash NP_{3s})/NP} \quad \frac{his\ mother.}{NP^{NP}}}{\frac{S\backslash NP_{3s}}{\lambda f.fj'} : \lambda x_1 \lambda x_2.love'x_1x_2 \quad \frac{NP^{NP}}{\lambda x_3.the-mother-of'x_3}} \\
 \frac{\frac{(S\backslash NP_{3s})/NP^{NP}}{\lambda g \lambda x.love'(gx)x}}{S\backslash NP_{3s} : \lambda x.love'(the-mother-of'x)x} \rightarrow \\
 \frac{S : love'(the-mother-of'j')j'}{} \rightarrow
 \end{array}$$

Notice that the result is a proposition, not a function. (I eschew as Jacobson does the analysis of English genitives.) If John loves somebody else’s mother,

then we would get the function S^{NP} as expected. I leave the mechanism and its implications for binding to much detailed discussion in Steedman (2011), Jacobson (1999).

The unary **Z** assumption carries with it some complications for VSO languages. For example, Welsh bound anaphora (27) might need syntactic wrap to apply (25) to the right argument, to the verb's category $S/NP_{3s}/_wNP$ to get $S/NP_{3s}/_wNP^{NP}$, where the slash subscript 'W' denotes wrap.

- (27) *Mi newidith* *Siôn ei feddwl.*
 PRT change.FUT.3s Sion 3MS mind.INF
 'Siôn will change his mind.' Welsh; Borsley, Tallerman and Willis
 (2007: 52)

Alternatively, we can consider another version of (25), viz. (28).⁶⁰ Its work is shown in (29) for the bound-pronoun interpretation.

- (28) $(X|_jNP)|_i Y: f \rightarrow (X|_i Y^{NP})|_j NP: \lambda x \lambda g. f x(gx)$ (z' -NP)

- (29) *Mi newidith* *Siôn* *ei feddwl*
 PRT change.FUT.3s Sion 3MS mind.INF
-
- $S/NP/NP_{3s}$ NP_{3s} NP^{NP}
 : $\lambda x_1 \lambda x_2. change' x_2 x_1$: s' : $\lambda z. the\text{-}mind\text{-}of' z$
-
- $S/NP_{3s}^{NP}/NP$
 : $\lambda x \lambda g. change'(gx)x$
-
- S/NP_{3s}^{NP}
 : $\lambda g. change'(gs')s'$
-
- S
 : $change'(the\text{-}mind\text{-}of' s')s'$
 'Siôn will change his mind.'

We are forced to get a free reading of 'his mind' from the individual-to-proposition interpretation of 'Siôn will change'. Its analysis is shown in (30). This string cannot be made a VP in any movementless theory—but it is indeed interpretable in CCG without extra devices, and it seems to suffice that it be a function so that individuals can take it as an argument to yield a proposition, via the $S \setminus (S/NP)$ type, or as a function to yield another function, via the $S^{NP} \setminus (S \setminus NP)^{NP}$ type.

$$\begin{array}{c}
 (30) \quad \frac{\frac{\frac{Mi \text{ newidith}}{\text{PRT change.FUT.3s}} \quad \frac{Si\acute{o}n}{\text{Sion}} \quad \frac{ei \text{ feddwl}}{\text{3MS mind.INF}}}{\frac{S/NP/NP_{3s}}{\lambda x_1 \lambda x_2. change' x_2 x_1} \quad \frac{NP_{3s}}{s'} \quad \frac{NP^{NP}}{\lambda x. the-mind-of' x}}}{\frac{S/NP}{\lambda x_2. change' x_2 s'}}}{\frac{S^{NP}/NP^{NP}}{\lambda g \lambda x. change'(gx)s'}}}{\frac{S^{NP}}{\lambda x. change'(the-mind-of' x)s'}}}
 \end{array}$$

We get the binding conditions that an anaphor inside the subject cannot be bound by object for free, if we assume the type-dependent solution to pronoun binding, rather than the structure-dependent solution of the familiar Chomskian kind, or Steedman-style LF as the level for binding. Given the NP^{NP} assumption for a pronoun, we cannot get a proposition (S) reading for the following example; it must at best be a function from things to propositions:

$$\begin{array}{c}
 (31)^* \frac{\frac{\frac{Prynodd}{\text{buy.PAST.3s}} \quad \frac{ei \text{ awdur } ei \text{ hun}}{\text{3MS author 3MS self}} \quad \frac{y \text{ llfyr.}}{\text{the book}}}{\frac{S/NP/NP_{3s}}{\lambda x_1 \lambda x_2. change' x_2 x_1} \quad \frac{NP_{3s}^{NP}}{s'} \quad \frac{S \setminus (S/NP)}{\lambda x. the-mind-of' x}}}{\frac{(S/NP)^{NP}/NP_{3s}^{NP}}{\lambda g \lambda x. change'(gx)s'}}}{\frac{(S/NP)^{NP}}{\lambda x. change'(the-mind-of' x)s'}}}
 \end{array}$$

*‘Its own author bought the book.’ Borsley, Tallerman and Willis (2007: 132)

In summary, the lexical type of a pronoun initiates, g projects, and z closes off the referential dependency of the bound pronoun, as in monadic computation. The process is an instance of threading the computation as $z(g)$, as Barker and Pryor (2010) showed. This is not the only monadic aspect of CCG, as we shall see in Chapter 10.

It seems possible, then, to find a purely type-dependent way to maintain Chomsky’s binding conditions as desiderata to narrow down the syntactic types, rather than add some conditions on a structured domain like LF. Therefore Steedman’s (2011) introduction of structure-dependence on the LF side,

on top of type-dependence in syntax-semantics correspondence, can be interpreted as a plea for computational parsimony in parsing, competence and its evolution, i.e. as a computational (read: *empirical*) challenge to cognitive science.

The counter-balance of the challenge is a long list of predictions we get from exponent types. For example: (a) syntactically differentiating the truly contextual pronoun binding versus its capture of an antecedent in syntax, so that for example an oracle can be called in to work depending on parser's output when the result is S^{NP} rather than S . (b) The empirically discernible distinction we get about the meaning of *John left* versus *he left*, as pointed out by Jacobson (1996).⁶¹ (c) The prediction of resumptive pronouns as possible lexical items, because we can systematically relate nonextraction categories like $(N \setminus N) / S^{NP}$ to extraction categories $(N \setminus N) / (S / NP)$. Note that the g -Z rule or the z -NP rule does not apply, hence these must be lexically mediated, which befits resumptive pronouns. (d) Can syntax *require* a pronoun? Jacobson's NP^{NP} type predicts that it may take part in the domain of locality of a construction.

I have no knowledge of such a finding, but the Welsh *cael* "get" passive comes close:

(32) *Cafodd Wyn ei rybuddio.*

Got.3s Wyn his warning

'Wyn was warned.'

Awbery (1976: 210)

Awbery (1976: 47) explains: "The passive sentence has a sentence-initial inflected form of *cael* (get) of the same tense and aspect as the verb of the active. This is followed by a noun phrase identical to the object of the active. Then comes a pronoun of the same person, number and gender (if it is 3sg) as this noun phrase, and an uninflected form of the verb in the active."

Awbery's data shows that what is dropped if the noun phrase after *cael* is a pronoun is the subject NP, not the possessive pronoun required by the passive:

(33) *Cawsom (ni) ein rhybuddio gan y ferch.*

Got.1pl (we) our warning by the girl

'We were warned by the girl.'

Awbery (1976: 48)

Therefore the pronoun is obligatory, and it is syntactically bound. It can be in the domain of locality of the head *cael*.⁶² The NP^{NP} type's relation to $NP|NP$ is predictable too. For example, Turkish headless relatives (34a–b) are indeed

pronominal, as the semantics implicated in the glosses show. They are derived from $(NP/NP)\backslash(S\backslash NP)$ of the relative participle which yields NP/NP for the relative clause, as in the headed variety (34c–d). (The examples are repeated from §2(11).)

- (34) a. $[[[İstanbul'a gid-en]_{NP/NP}]_{-ler-i}]_{NP} NP$ *ben gör-me-di-m.*
 Ist-DAT go-REL-PLU-ACC I see-NEG-PAST-1s
 ‘I did not see the ones that go to Istanbul.’
- b. $[[[İstanbul'a git-tik]_{NP/NP}]_{-ler-im}]_{NP} NP$ *daha güzel-di.*
 Ist-DAT go-REL-PLU-POSS.1s more beautiful
 ‘The ones with which I went to Istanbul looked better.’
- c. $[İstanbul'a gid-en]_{NP/NP}$ *otobüs*
 Ist-DAT go-REL bus
 ‘The bus that goes to Istanbul’
- d. $[İstanbul'a git-tiğ-im]_{NP/NP}$ *otobüs*
 Ist-DAT go-REL.1s bus
 ‘The bus with which I went to Istanbul’

To recapitulate: employing the combinators for variable-free semantics does not seem to violate the transparent import of order-instigated semantics of combinators to their syntacticization. Doing without them forces us to make auxiliary assumptions. Moreover, some constituents seem to show asymmetric behavior regarding the exponent types. I exemplify some of them in the next section. These are new research agenda for the entire family of CCG models.

4. More on LF: Unary BCWZ, constituency and coordination

In an LF-less system, we not only need Jacobson’s (1999) unary **Z** but unary **B** as well, to account for multiple pronouns and their binding possibilities. The first example below is obtained if the verb *said* undergoes unary **Z** first and then unary **B**, as Jacobson (1999) showed. We get the second example if the order is reversed.

- (35) a. *Every man_i thinks that every boy_j said that his_j mother loves his_i dog.*
 (Jacobson 1999)

- b. *Every man_i thinks that every boy_j said that his_i mother loves his_j dog.*

Recall the unary **B**'s devastating effects on complete constituents, repeated here:

- (36) a. *I think that Wittgenstein might have liked Kafka.*

$$\frac{\frac{VP/S'}{S'/S_{fin}} \quad \frac{S_{fin}}{that} \quad \frac{S_{fin}}{might\ have\ liked\ Kafka}}{\frac{S'/S_{fin}}{(S'\backslash NP)/(S_{fin}\backslash NP)} \quad \frac{S_{fin}\backslash NP}}{S'\backslash NP} \text{---app}}{S'} \text{---<}$$

Jacobson's account avoids this problem by keeping the complete constituents complete albeit a unary **B**: the word *that* undergoes a type-shift to S^{NP}/S_{fin}^{NP} by (*g-NP*) to eliminate (36b).

Likewise, Szabolcsi's use of unary **BCW** avoids deriving a nonconstituent, by building them into the lexical categories. Therefore a **BTS** binary core syntax seems uncontroversial, except for Shaumyan (1977, 1987)-style combinatory semantics where two expressions are related by combinators, for example *that man I hate him* and *I hate that man* by **K**. That seems to have a different agenda than a search for a radically lexicalized adjacency system for grammar.

Thus the theoretical differences come down to the interpretation of some empirical issues, repeated below: (i) *He lost* in (37a) is considered *S* by variable-friendly semantics and S^{NP} by variable-free, (ii) the asymmetry of binding in (37b–c) are attributed to LF conditions in variable-friendly systems and to lexical generalizations about arguments in variable-free, and (iii) the lack of respect to LF conditions in nonlocal constructions in (37d–e) and in relativization are handled by the conspiracy of lexical syntactic and semantic types in either view. (37f–g) are still divisive, as pointed out earlier.

- (37) a. *Every man_i thinks (that) he_i lost and (that) Mary won.* Jacobson (1999)
 b. **Sheself left.* Szabolcsi (1992)
 c. **Sheself sees everyone.*
 d. *A violin_i which this sonata_j is easy to play_j on_i*
 e. **A sonata_i which this violin_j is easy to play_i on_j*

- f. *Every man_i thinks that every boy_j said that his_j mother loves his_i dog.*
- g. *Every man_i thinks that every boy_j said that his_i mother loves his_j dog.*

The exponent vocabulary for syntactic and semantic types therefore creates not two incommensurate categorial landscape (that would be the case for wrap systems), but some degree of freedom.

The treatment of (37a) bears on constituency in an indirect way, in the result categories of coordination, precisely because opinion is divided about the category of *He lost* and about the nature of extraction in resumptive pronouns. Consider the examples below.

(38) a. *Every man_i loves and no man_j marries his_{i&j/*i/*j} mother.*
 b. *Every man_i thinks he_i lost and Mary won.*

$$\begin{array}{c}
 \overline{NP^{NP}} \quad \overline{S \setminus NP} \quad \overline{(X \setminus X) / X} \quad \overline{S} \\
 \overline{S^{NP} \setminus NP^{NP}} \quad \overline{S \setminus S} \\
 \overline{S^{NP}} \quad \overline{S^{NP} \setminus S^{NP}} \\
 \overline{S^{NP}}
 \end{array}$$

As Jacobson (1999) points out, the NP^{NP} type for pronouns maintains (a) the across-the-board CSC asymmetry without extra assumption, that it is impossible to bind *out* of one conjunct in (38a), and possible to bind into just one in (38b), and (b) that the “like-category constraint” for CSC is not enough if we do not make the three-way $\{S, S^{NP}, S|NP\}$ distinction.

The derivation in (38b) maintains the “like category” explanation for coordination without extra assumption. It is not a violation of application-only modality of the coordinator *and*, because no new slashes are introduced by $g-NP$. We shall see in monadic computation (Chapter 10) that the slash in unary composition of (22) can indeed be without modality.

Regarding the asymmetry in coordination in relation to pronominal reference, we can look at the rightward conjuncts with functions rather than propositions. Interesting possibilities arise in a modalized CCG. Jacobson’s suggestion of unary composition might appear to make coordinands susceptible to island violations, but it does not. We can maintain the islandhood of conjuncts by disallowing composition into them using the application-only modality. Jacobson’s (1999) suggestion to type-raise the S of leftward con-

junct to $S/(S\backslash S)$ to derive (39a) avoids the composition of *Mary won* with *and* (39b).

$$\begin{array}{c}
 (39) \text{ Every } man_i \text{ thinks } \quad \textit{Mary won} \quad \textit{and} \quad \textit{he}_i \text{ lost} \\
 \hline
 \begin{array}{ccc}
 \frac{S/(S\backslash S)}{g-NP} & \frac{(X\backslash_*X)/_*X}{g-NP} & \frac{S^{NP}}{S^{NP}} \\
 \hline
 S^{NP}/(S\backslash S)^{NP} & (X\backslash_*X)^{NP}/_*X^{NP} & \\
 \hline
 & (S\backslash_*S)^{NP} & \rightarrow \\
 \hline
 & S^{NP} & \rightarrow
 \end{array} \\
 \\
 \begin{array}{c}
 (a) \\
 *Every \text{ man}_i \text{ thinks } he \quad \textit{Mary won} \quad \textit{and} \quad \textit{would lose.} \\
 \hline
 \begin{array}{ccc}
 \frac{S/(S\backslash S)}{g-NP} & \frac{(X\backslash_*X)/_*X}{g-NP} & \frac{S\backslash NP}{g-NP} \\
 \hline
 S^{NP}/(S\backslash S)^{NP} & (X\backslash_*X)^{NP}/_*X^{NP} & S^{NP}\backslash NP^{NP} \\
 \hline
 & \begin{array}{c}
 *** \\
 \rightarrow B \\
 S^{NP}/S^{NP}
 \end{array} & \\
 \hline
 (b)
 \end{array}
 \end{array}
 \end{array}$$

In summary: in exploiting the degrees of freedom afforded by exponent types of Jacobson, lexical generalizations of combinators and variable-friendly logical forms, we are within the program of radical lexicalization. The unary combinatory rules have substantive constraints on them, or they are built into the lexical categories. In other words, they are lexical rules. No combination rule or lexical rule depends on LF in systems where it is posited as a level. The empirical coverage of constituency is the same, although some empirical assumptions, theoretical choices and predictions differ.

Variable-free semantics spells a tightly controlled unary system with an interlocking choice of constraints on for example pronouns, different kinds of verb classes, reflexives, relative pronouns, object categories etc. Its highly nondeterministic type-shifting rules seem to add no more burden than the result that type-raising must operate as a universal rule anyway; it cannot be fully lexicalized. Its use of model-stage storage to take care of quantifier scope as done by Cooper (1983) does require another stack, but, as long as that stack does not interact with the parser's category stack, having two stacks does not automatically give us Turing-completeness or a more liberal computation.⁶³ On the other side, variable-friendly semantics of the LF kind is forced to posit a model theory over and above what the standard logics provide, such as for example in Steedman (2011).

In both cases, surface compositionality is maintained, for a good reason. It appears that the logician's logical form is the cognitive scientist's and computational linguist's predicate-argument structure and dependencies. The notion of LF is entirely uncontroversial in computational linguistics, to the extent that it is almost always implicitly assumed, because otherwise the task of using a grammar in both ways to parse and generate is unreasonably complicated.

This LF is in most cases not Chomsky's or May's LF, because no predication over such a level is bothered to be checked in the first place. Noise in the data (ambiguity, vagueness, misunderstanding, misperception, misconception, misattention, misaction etc.) far outweighs the noise that might be introduced by not checking the LF conditions on the hypotheses.

Cognitive scientists with a computational bend use LF as an approximation of PADS in learning syntactic categories from PF-PADS pairs where the category is the hidden variable (after all, it is not observable). To go from models to PADS in that task is complex, and the search space for the hidden variable is much less constrained.

Recall also that the Condition A-like innate knowledge, that children never entertain the possibility of e.g. **sheself*, can be subsumed by a conspiracy of universal constraints on the lexicon: (a) that all arguments are type-raised, (b) argument-taking is combinatory knowledge (e.g. knowledge of **W** dependency presumes knowledge of curried transitivity, which also brings in coargumenthood without further assumptions), (c) lexicalizable variables—pronouns—are not semantic variables but unknowns.

A linguistic representation of semantics can be an uncontroversial assumption, independent of whether we posit a Steedman-style LF without extra syntax, a Pesetsky (1985)-style LF with its own syntax, or a Montague-style derivation structure where some scope bookkeeping is sufficient for a model-theoretic interpretation.

This LF is linguistically interesting to the extent it represents or models asymmetries, such as scope and binding. There is no language with a subject reflexive.⁶⁴ Logically it seems perfectly possible, as say $(\forall x)(x=mary' \Rightarrow see'xx)$, which would be a legitimate logical representation for *Mary saw herself*, as well as for **sheself saw Mary*, and **Herself saw Mary*.

Some striking counterexamples to this long-standing observation have been shown by Postal and Ross (2009). English, Albanian and Greek inverse reflexives, which are the least oblique (subject) reflexives with clausemate antecedents, strengthen the need for a linguistic representation because they

require, according to Postal and Ross, the notion of *derived subject*, a strictly linguistic concept, as in Relational Grammar (see Blake 1990 for RG concepts).

Consider another case for a linguistic representation. The Turkish plural marker must be considered polysemous if we want to eschew an LF representation. We have (40a), in addition to the nonlocative extensional interpretation of the plural (40b).

- (40) a. *Yarın akşama Ahmet'lere davetliyim.*
 Tomorrow night-DAT A-PLU-DAT invited-1s
 'I am invited to Ahmet's for tomorrow night.' Turkish
- b. *Kendini kitaplara verdi.*
 self-ACC book-PLU-DAT give-PAST
 'S/he gave himself/herself to the books.'

The expression in (40a) is three-way ambiguous: (1) There may be more than one people at Ahmet's, with Ahmet being the representative of the group, (2) there might be only Ahmet at Ahmet's, or (3) there might be somebody else, or even no one, at Ahmet's. In the last case the speaker would know the place as Ahmet's, just as s/he would know Mehmet's, Ayşe's, Mary's as places, thanks to the plural. The first reading is closest to an extensional interpretation of the plural, but the other two are intensional. That kind of polysemy-turned-ambiguity might render the idea of radical lexicalization vacuous, because any marker can be intensional or extensional in this regard:

- (41) a. *dünyanın tepesi*
 world-GEN top-POSS
 'the top of the world' Turkish
- b. *adamin arabası*
 man-GEN car-POSS
 'the man's car'

A Montague-style intensional logic (IL) has room to work from a type say *plu,*' but the core translation of Montague's IL is disambiguated, therefore we would need two types or two rules to intensionalize and extensionalize the plural. A PADS presentation could have one entry to be mapped to Montague's intensional-extensional world. Partee and Rooth (1983) show how type-shifting can relate one grammatical object with many model-theoretic objects.

In regard to the combinatory syntactic knowledge of plurality, there is no distinction between the intensional and extensional interpretation, hence we would expect a single category. As a knowledge of the full interpretability of a meaning-bearing element, we can conceive a two-way IL translation both of which are disambiguated, or use Partee and Rooth idea to define a function from one PADS object to a powerset of a finite set of types, which would also secure a lexical representation along with PADS. This does not directly relate to meanings out there but to model-theoretic constraints on PADS objects like *plu*,¹ hence it can be considered part of competence because it is linked to PADS, which is an essential part of a category. The noncommittal view of PADS toward truth conditions is also defended on the following grounds.⁶⁵

Language embodies no particular metaphysics; it embraces both Realism and Psychologism. However, psychology has the last word. Whatever the semantics of a term, its relation to the world depends on human cognitive capacity. A word with a Realist semantics would only be coined or maintained in use by virtue of its associated mental schema. Likewise, whatever the semantics of a term, it is not mentally represented in isolation. Johnson-Laird (1983: 204)

The narrow research program pursued here is that, whatever the nature of representation of semantics is, it must relate to syntax compositionally, because it is one end of the syntactic process. Whether it spells a truth-conditional semantics or some kind of mental and social world of thoughts and concepts is implicated here to be an interface issue; see Chapter 9, in particular §9.3 and §9.10, for further discussion. The topic is an open debate in cognitive science; witness a recent target article of Feldman (2010) and subsequent discussion in the same volume, with responses and criticism by Allen, Partee, Steels and Steedman.

Chapter 7

Further constraints on possible grammars

A CCG grammar is a finite set of lexicalized category assignments to strings. The language of the grammar is its closure on the invariants listed in Table 2. Thus everything projects from the lexicon, because the invariants do not encode any language-specific information. It follows that all substantive constraints must be enforced on the lexicalized syntactic types, because the syntactic process is completely syntactic type-driven.

A lexical category must therefore capture all the syntactic and semantic dependencies as knowledge of that string, say a word, since no other knowledge can be added during the syntactic process, and none deleted.

Steedman offers the following principle as a constraint on possible categories.

(1) The Principle of Categorial Type Transparency: (PCTT)

For a given language, the semantic type of the interpretation together with a number of language-specific directional parameter settings uniquely determines the syntactic category of a category. Steedman (2000b: 36)

The principle works both ways (Steedman calls syntax-to-semantics mapping the inverse of (1)). The semantic type of an interpretation is entirely determined by the syntactic type:

- (2) Take T to be the type relation with an inverse. If α has the syntactic type A and β type B , then $T(\alpha, \beta) = T\beta \mid T\alpha = B \mid A$, for some '|'. If (α, β) has a basic type A , then $T(\alpha, \beta) = A$. Inversely, $T^{-1}(B \mid A) = (T^{-1}A, T^{-1}B) = (\alpha, \beta)$, for $A_{(\alpha)}$ and $B_{(\beta)}$. $T^{-1}(A) = \alpha$ for a basic type $A_{(\alpha)}$.

For example, assume the following types for English.

- (3) $S: t$ *Kafka died.*
 $S: (e, t)$ *Kafka adored*
 $NP: e$ *Kafka*
 $N: (e, t)$ *man*

Given these types, $S \setminus NP$ can be (e, t) (functions onto propositions), or $(e, (e, t))$ (functions onto functions, where for example the result function

wants a discourse participant). We need not eliminate the second variety from theory (perhaps we cannot), when experience can sort it out. The S/NP can be $(e, (e, t))$ (functions onto predicates), or (e, t) (functions onto propositions, where for example the subject of the action is implicit, say *self'*). In the last case, we can safely assume that the implicit participant is not the syntactic object, because English subjects are not compatible with $'/NP's$, therefore that NP must be the object.

The principle suggests that, given these English-specific pairs, a category such as $S/(S\backslash NP)$ cannot be anything other than $((e, t), t)$ if S is t , and a category such as $S\backslash(S/NP)$ can only be $((e, (e, t)), (e, t))$ if S is (e, t) .

PCTT is a relation, not a function with an inverse. For example, it is entirely possible that nominals get two categories in a language, say $NP: e$ (proper names), and $NP: (e, t)$ (properties). Then $S\backslash NP$'s semantic type can be (e, t) or $((e, t), t)$. What it does not allow is this: if X is of type α and $Y \beta$, then $X|Y$ cannot be anything other than (β, α) . Given a lexical pair of types, they are functionally dependent on each other.

Take for example $N: \lambda x.man'x$ and $S\backslash NP: \lambda x.sleep'x$. The x of *man'* is not a syntactic variable. We can deduce this property from the semantic type of *man,'* which is (e, t) . The x of *sleep'* must be a syntactic variable, which corresponds to the $'\backslash NP'$ of $S\backslash NP$. Thus lambdas are not nominally designated as syntactic or semantic. These properties follow from their lexicalized syntax translated from dependency semantics via adjacency. N cannot have a syntactic argument glued (by $'\cdot'$) to a semantic object. $S\backslash NP$ cannot take place in syntax without a syntactic argument glued to its participant role.

Jacobson's (1999) pronouns, and proposition versus function distinction of S can be covered by PCTT as well. Assuming (e, e) for NP, NP as she does, we are forced to an (e, t) interpretation of S, NP where the e is not a syntactic argument, because the syntactic type is not $S|NP$. Since PCTT is not a function, we are not forced to assume that an S is always t type (that possibility would rule out a function interpretation of S , such as functions from individuals to propositions as in pronouns). It can be (e, t) .

The use of lambdas as the glue language of the $'\cdot'$ relation in syntax-semantic correspondence therefore depends on the semantic types. Eta-normalization can eliminate variables from (e, t) types of various syntactic functions, e.g. from $N: \lambda x.man'x$ and $S\backslash NP: \lambda x.sleep'x$, which reveals the explicit role of the slash in syntactic argument-taking as a reflection of semantic argument-taking. The potential confusion about whether lambdas are

syntactic or semantic abstractions can be avoided if we use typed objects all the time, for example to claim that *sleep'* is a one-argument syntactic function which also happens to be a one-argument semantic function, and *man'* is a zero-argument syntactic function which is a one-argument semantic function.

Using adjacency formulations of argument-taking over strings makes the distinction explicit. The Schönfinkel-Curry arity of *man* is *man'*, i.e. zero. The arity of *sleep'* is 1, from $\mathbf{B}^1 \mathbf{I} \textit{sleep}'$.⁶⁶

Thus the number of syntactic lambdas in the glue language is the power of \mathbf{B} in a semantic object's prefix. It is the same as the number of argument slashes in the syntactic type, and no confusion arises.

With PCCT we can eliminate types such as (4) from the space of possible categories, hence possible grammars.

- (4) a. $*\textit{sleep} := S : \lambda x. \textit{sleep}'x$
 b. $*\textit{sleep} := (S \setminus NP) / NP : \lambda x. \textit{sleep}'x$

The first example says that all sleeping is syntactically memorized, because it does not take any syntactic arguments, yet its semantics might suggest that (a) it does take a syntactic argument since it is a reflection of $\mathbf{B}^1 \mathbf{I} \textit{sleep}'$, or (b) it is a function, in which case what it is a function of is not clear since the syntactic type is not S^X for some X . If it is a property named *sleep*, as in *sleep causes absenteeism*, then it would be fine but inconsistent with other properties, which are usually of type N or NP , but not S . Only cross-situational learning can remedy this problem, therefore the argument role/property interpretation must be considered legitimate.

The second example (4b) does not claim that *sleep'* cannot be a transitive verb. PCTT and its combinatory origin (Schönfinkel-Curry arity) simply say that if it is, then there must be another lambda, otherwise this category cannot be construed as the knowledge of the word.

Thus the system is conditional on the current assumptions about the syntactic reflection of states of affairs, and needs no universal base such as in Jackendoff (1997) or Hopper and Thompson (1980) (the latter work assumes transitivity is universal). There can be a ditransitive *sleep* predicate as far as CCG is concerned, a fact which we must be able to discern from its syntactic behavior.

The syntactic lambdas and the semantic ones can be eliminated by eta-reduction as we have seen. What cannot be eliminated are the structural unknowns of the Logical Form (LF), if we follow the LF-friendly combinatory

path. In that sense, Steedman's unknowns are not the kind of objects that Schönfinkel's combinators are designed to eliminate.

Steedman (2000b) offers two more substantive principles, the Principle of Lexical Head Government (PLHG), and the maxim of Head-Categorial Uniqueness (HCU). The first principle amounts to saying that lexical categories must not proliferate just because there are many syntactic contexts in which a lexical item can take part, such as the word *chews* in the examples below, among others.

- (5) a. *The cat chews the mat.*
 b. *The cat chews itself.*
 c. *the mat which I believe the cat chews*
 d. *The cat chews and the dog scratches the mat.*
 e. *This mat the cat chews all the time.*

By the same principle, the passive in *the mat was chewed by the cat* and the infinitive in *the cat wants to chew the mat* involve the same lexical item, namely *chew*. These principles do not reduce the space of possible categories, but they do put constraints on individual grammars, which makes the size of a grammar a meaningful number. McConville (2006) makes use of this number to choose among potential competence grammars.

The principles we have covered so far bear on lexical correspondences, and they reduce the space of possible grammars because by the radical lexicalization of the rule-to-rule hypothesis, a particular grammar can only be read off the lexical syntactic types. We shall see in §9.7 that the theory of functional categories employed in transformational grammar can also be seen as providing further constraints on possible syntactic types. The reason why it is considered a meta-theory for CCG is because functional categories do not seem to arise from combinatory dependencies, therefore not from a combinatory manifestation of adjacency. For example, $\lambda P.Pa'$ can characterize both syntactic subjects and syntactic objects with semantics a' . Their differences in agreement and finite domains must arise from differences in the syntactic features of basic categories in a syntactic type. PCTT can only partially help in these matters, such as distinguishing $S/(S \setminus NP)$ and $S \setminus (S/NP)$, so that a theory of agreement or binding can make use of the distinction.

Szabolcsi's (1989, 1992) constraints on the lexicon narrow down the possible lexical categories, hence, by radical lexicalization, possible grammars.

We can also think of other kinds of substantive constraints on possible grammars, some of which need not worry a grammar theorist. For example,

what could stop a group of people from acquiring a language in which every sentence ends with the same word? A linguistic theory would be overextending itself in trying to address such matters when experience can sort it out. It might be in the Zipfian tail of possible languages.

Chapter 8

A BT_{SO} system

What can be the syntactic roles of the combinators other than **BTSCWZ**? I list the remaining set below, along with their equivalences:

- (1) **Y** $Yx = y = xy$ for some y depending on x
Φ $\Phi_{xyzw} = x(yw)(zw)$ $\Phi = \mathbf{B}(\mathbf{BS})\mathbf{B}$
Ψ $\Psi_{xyzw} = x(yz)(yw)$ $\Psi = \mathbf{B}(\mathbf{BW}(\mathbf{BC}))(\mathbf{BB}(\mathbf{BB}))$
J $J_{xyzw} = xy(xwz)$ $J = \mathbf{B}(\mathbf{BC})(\mathbf{W}(\mathbf{BC}(\mathbf{B}(\mathbf{BBB}))))$
O $O_{xyz} = x(\lambda w.y(zw))$ $O = \mathbf{C}(\mathbf{BBB})\mathbf{B}$

Recall that $\mathbf{C} = \mathbf{B}(\mathbf{T}(\mathbf{BBT}))(\mathbf{BBT})$, and $\mathbf{W} = \mathbf{ST}$. Thus with the exception of **Y**, they must be lexicalized in a **BTS** system, according to Szabolcsi's criterion in §5(38). We have seen in §4.1 that **Y** is not finitely typeable, hence its finite representability cannot be assumed. Let us look at the finitely typeable ones. I leave out **J** because, as explained in §4.4, its behavior has not been observed in any language.

Recall also that Szabolcsi's hypothesis is not sufficient to rule out **K** and **I** from syntax. It is a formal restriction. We needed empirical support to eliminate **K** and **I**. We also needed empirical support to suggest why **B** and **S** must operate binarily and not ternarily, which was also not covered by her hypothesis. These efforts can be considered as investigating the empirical import of Schönfinkel's fully binarized function-argument notation (currying), an otherwise formal result.

Take for example **Φ** and **O**. **Φ**'s semantics is that of coordination. The formal criterion suggests that it is lexicalizable because $\Phi = \mathbf{B}(\mathbf{BS})\mathbf{B}$. Empirically it is clear that coordination is lexicalized in languages, because there are languages which do not have syntactic coordination, for example Hixkaryana (Derbyshire 1979) and Dyirbal (Dixon 1972). And, every coordinating language seems to have a lexical head for it (*and*, *but* etc.), or restrict it to certain tunes. That is, there is always some syntactic object even if it is not a word to which we can assign the semantics of coordination in the lexicon, in the manner of Steedman (2000a). Therefore both formal and empirical results suggest that **Φ** must be a lexicalized combinator.

Not so for **O**. By the formal criterion (§5(38)), it can be lexicalized because $O = \mathbf{CB}^2\mathbf{B}$, and **C** is definable by **B** and **T**. Empirical facts suggest oth-

erwise. Recall that unlike other combinators, **O** is a combinator but not a supercombinator. This is evident in its definition $\mathbf{O}fgh = f(\lambda x.g(hx))$, with its unmovable inner lambda abstraction: x is not an argument of **O**.

This combinator seems to be at odds with lexicalization when we consider that we are facing **O** semantics in strings such as *what you can* (2), which seems not to be lexicalized, for example *what you can and what you should not do*.

$$\begin{array}{c}
 (2) \quad \frac{\frac{\frac{\textit{what}}{S/(S/NP)} \quad \frac{\textit{you}}{S/(S \setminus NP)} \quad \frac{\textit{can}}{(S \setminus NP)/(S \setminus NP)}}{\lambda Q. ?y Qy : \lambda f.f \textit{you}' : \lambda P \lambda x. \textit{can}'(Px)} \quad \text{>B}}{S/(S \setminus NP)} \quad \text{>B}}{\lambda P. ?y \textit{can}'(P \textit{you}')} \quad \text{>B}}{S/((S \setminus NP)/NP)} \quad \text{>O}}{\lambda P. ?y \textit{can}'(P \textit{you}')} \quad \text{>O}
 \end{array}$$

Does this justify the incorporation of **O** into syntax? Recall the syntacticization of binarized **O**, which is at work in (2):

$$(3) \quad X/(Y/Z) : f \quad Y/W : g \rightarrow X/(W/Z) : \lambda h.f(\lambda x.g(hx)) \quad ({}_2\mathbf{O})$$

Hoyt and Baldrige (2008) provide the following examples from various languages which cannot be handled by a **BTS** system, a result which suggests free operation in syntax. They call such constructions *cross-conjunct extraction*, first noted by Pickering and Barry (1993). All bracketed strings in these examples arise from syntactic and semantic assumptions similar to (2).

- (4) a. .. [What you can] and [what you must not] base your verdict on
 b. [dat ik haar wil] en [dat ik haar moet] helpen
 that I her want and that I her can help
 ‘..that I want to and that I can help her.’ Dutch
 c. [Wen kann ich] und [wen darf ich] noch wählen?
 who can I and who may I still choose
 ‘Whom can I and whom may I still choose?’ German
 d. Gandes-te [cui ce] vrei,
 consider-IMP.2s-REF.2s who.dat what want.2s
 și [cui ce] poți, să dai.
 and who.dat what can.2s to give.SUB.2s
 ‘Consider to whom you want and to whom you are able to give
 what’ Romanian

- e. [*Me lo puedes*] y [*me lo debes*] *explicar*
 me it can.2s and me it must.2s explain
 ‘You can and should explain to me’ Spanish

But, as they note, the same effect can be achieved by having multiple categories for function words because these kinds of semantic dependencies are headed by them. The Turkish facts lead to the same conclusion: it is the relative pronoun that seems to engender such kinds of constituencies.

- (5) a. *Ben-im uyu-ma-diđi-ni* [*savun-duđum*] ve [*ispat et-tiđim*] *şoför*
 I-1ssleep-NEG-COMP-ACC defend-REL.1s and proof do-REL.1s driver
 ‘The driver who I claimed and proved that s/he did not sleep.’

- b. **Ben-im uyu-ma-diđi-ni* [*savun-duđum*] ve [*ikna ol-duđum*] *şoför*
 persuade be-REL.1s

- c.
$$\frac{\frac{\textit{savun}}{S \backslash NP_{\text{agr}} \backslash S'_{\text{acc}}} \quad \textit{-duđ-um}}{(NP/NP) \backslash NP' \backslash (S \backslash NP \backslash NP)}}{(NP/NP) \backslash NP' \backslash (S'_{\text{acc}} \backslash NP)} \textcircled{\text{O}}$$

The crucial step that distinguishes (5a–b) is shown in (5c). It is the backward variety of (3). The verb *ikna* ‘persuade’ requires a dative-marked nominalized clause therefore it cannot yield a like-category with *savunduđum*, which needs an accusative-marked complement clause. This information is transparently projected by **O**.

- (6) $Y \backslash W : g \quad X \backslash (Y \backslash Z) : f \rightarrow X \backslash (W \backslash Z) : \lambda h.f(\lambda x.g(hx))$ (2**O**)

Example (5c) might appear to suggest that the derivation can be lexicalized because a phonological word is syntactically derived, but the coordination data such as (5b) and (7) show that what takes place is indeed syntax:

- (7) *Ben-im dava-sı-ni* [*bil-ip savun*]-*duđum adam*
 I-1s law suit-POSS.3s-ACC know-CONV defend-REL.1s man
 ‘The man whose lawsuit I knew and which I defended.’

The extra categories which allow us to lexicalize the **O** semantics in these examples are not well motivated in English or Turkish. Take for example the category $S/(VP/NP)/(S/NP)$ for *what*, which Hoyt and Baldrige (2008) rightfully consider doubtful, in addition to its well-motivated category $S/(S/NP)$. The last category is empirically sound, as shown in (8a–b), but the extra category is not always sound (cf. 8c–d). Thus attempts to keep such data under the **BTS** syntax by lexicalizing the **O** are not very convincing.

- (8) a. $\frac{\frac{\text{What}}{S/(S/NP)} \quad \frac{\text{did John hit?}}{S/NP}}{S} \text{app}$
- b. $\frac{\frac{\text{What}}{S/(S/NP)} \quad \frac{\text{you can and what you must not do}}{S/VP}}{S/(VP/NP)} \text{o} \quad \frac{\text{VP/NP}}{VP/NP}$
- c. $\frac{\frac{\text{What}}{S/(VP/NP)/(S/NP)} \quad \frac{\text{did John hit?}}{S/NP}}{?? S/(VP/NP)} \text{app}$
- d. $\frac{\frac{\text{What}}{S/(VP/NP)/(S/NP)} \quad \frac{\text{you can and what you must not do}}{S/NP}}{S/(VP/NP)} \text{app} \quad \frac{\text{VP/NP}}{VP/NP}$

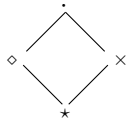
We know that **O** does not satisfy Szabolcsi's formal criterion for free operation in a **BTS** system, because $\mathbf{O} = \mathbf{C}(\mathbf{B}^2)\mathbf{B}$, and $\mathbf{C} = \mathbf{B}(\mathbf{T}(\mathbf{BBT}))(\mathbf{BBT})$. We also know that adding **O** to syntax would not change the automata-theoretic results because of the possible formulation of **O** by **B** and **T** as above; Vijay-Shanker and Weir's (1994) argument for linear-indexed behavior of CCG makes crucial use of these combinators, and only these combinators.

In summary, lexicalizing the **O** because of these concerns poses an empirical problem to a CCG lexicon, and ignoring the **O**-constituents would mean a loss of empirical coverage in syntax.

The binary **O** is not redundant in a system of binary **B**, binary **S** and the finite powers of **B**. Let us look at the formulation of **O** without **C** to see this result. $\mathbf{O} = (\mathbf{B}(\mathbf{T}(\mathbf{BBT}))(\mathbf{BBT}))(\mathbf{B}^2)\mathbf{B}$. Although binary **B** is at work in this definition, it also needs unary **T**, unary **B** and unary **B**², to yield the **O**-semantics for adjacent substrings ω_1 and ω_2 . Thus the **O**-constituents need the binary **O** because some of these combinators are not freely operating.

The **BTSO** system which emerges from these considerations is listed in Table 2. I suggest the name *orifice* for **O** to symbolize its 'leaking lambda' inside the dependencies. All possible directional-modal alternatives of combinators are listed for completeness. Only small powers are presented to save space. Since any lexicon is bounded by a maximum number of arguments, say n , we can take the required power to be $m-1$ where m is the maximum of such n among possible languages, which is by definition some number, rather than a variable. Steedman (2000b) suggests $n=4$ for English.

Table 2. The syntacticized **BTSO** system.

Application	$X/_*Y$	Y	\rightarrow	X	$>$
	Y	$X \backslash_* Y$	\rightarrow	X	$<$
Composition	$X/_\diamond Y$	$Y/_\diamond Z$	\rightarrow	$X/_\diamond Z$	$> \mathbf{B}$
	$Y \backslash_\diamond Z$	$X \backslash_\diamond Y$	\rightarrow	$X \backslash_\diamond Z$	$< \mathbf{B}$
	$X/_\times Y$	$Y \backslash_\times Z$	\rightarrow	$X \backslash_\times Z$	$> \mathbf{B}_\times$
	$Y \backslash_\times Z$	$X \backslash_\times Y$	\rightarrow	$X \backslash_\times Z$	$< \mathbf{B}_\times$
	$X/_\diamond Y$	$(Y/_\diamond Z) W$	\rightarrow	$(X/_\diamond Z) W$	$> \mathbf{B}^2$
	$(Y \backslash_\diamond Z) W$	$X \backslash_\diamond Y$	\rightarrow	$(X \backslash_\diamond Z) W$	$< \mathbf{B}^2$
	$X \backslash_\times Y$	$(Y \backslash_\times Z) W$	\rightarrow	$(X \backslash_\times Z) W$	$> \mathbf{B}_\times^2$
	$(Y \backslash_\times Z) W$	$X \backslash_\times Y$	\rightarrow	$(X \backslash_\times Z) W$	$< \mathbf{B}_\times^2$
Type Raising		A	\rightarrow	$T /_i (T \backslash_i A)$	$> \mathbf{T}$
		A	\rightarrow	$T \backslash_i (T /_i A)$	$< \mathbf{T}$
Substitution	$(X/_\diamond Y) /_\diamond Z$	$Y /_\diamond Z$	\rightarrow	$X /_\diamond Z$	$> \mathbf{S}$
	$Y \backslash_\diamond Z$	$(X \backslash_\diamond Y) \backslash_\diamond Z$	\rightarrow	$X \backslash_\diamond Z$	$< \mathbf{S}$
	$(X \backslash_\times Y) \backslash_\times Z$	$Y \backslash_\times Z$	\rightarrow	$X \backslash_\times Z$	$> \mathbf{S}_\times$
	$Y \backslash_\times Z$	$(X \backslash_\times Y) \backslash_\times Z$	\rightarrow	$X \backslash_\times Z$	$< \mathbf{S}_\times$
	$(X/_\diamond Y) Z$	$(Y /_\diamond W) Z$	\rightarrow	$(X /_\diamond W) Z$	$> \mathbf{S}''$
	$(Y \backslash_\diamond W) Z$	$(X \backslash_\diamond Y) Z$	\rightarrow	$(X \backslash_\diamond W) Z$	$< \mathbf{S}''$
	$(X \backslash_\times Y) Z$	$(Y \backslash_\times W) Z$	\rightarrow	$(X \backslash_\times W) Z$	$> \mathbf{S}_\times''$
	$(Y \backslash_\times W) Z$	$(X \backslash_\times Y) Z$	\rightarrow	$(X \backslash_\times W) Z$	$< \mathbf{S}_\times''$
Orifice	$X /_\diamond (Y Z)$	$Y /_\diamond W$	\rightarrow	$X /_\diamond (W Z)$	$> \mathbf{O}$
	$Y \backslash_\diamond W$	$X \backslash_\diamond (Y Z)$	\rightarrow	$X \backslash_\diamond (W Z)$	$< \mathbf{O}$
	$X \backslash_\times (Y Z)$	$Y \backslash_\times W$	\rightarrow	$X \backslash_\times (W Z)$	$> \mathbf{O}_\times$
	$Y \backslash_\times W$	$X \backslash_\times (Y Z)$	\rightarrow	$X \backslash_\times (W Z)$	$< \mathbf{O}_\times$
Legend:	$>$	forward			
	$<$	backward			
	$> \Sigma_\times$	forward crossing Σ			
	$< \Sigma_\times$	backward crossing Σ			
	A	argument types of class of values T			
T	value types of class of arguments A				
			Modalities:		

It is a prediction of CCG that all these rules can be potential mergers in some language. They are not different in kind because they arise from currying and the adjacency of combinators, but they all manifest a different kind of syntacticized semantic dependency, including directionality. Thus the explanation offered by CCG is that syntax can be a reflex-like process because nothing needs to be remembered in the construction of constituency or interpretation—i.e. in parsing—when all the possible dependency projections are factored into the universal rules. Thus every word and phrase projects its syntax and semantics onto surface constituents, and they do not fall prey to some grammar-external constraint when taking part in syntax.

We have seen the harmonic composition rules and some substitution rules at work. Below I exemplify the crucial involvement of most of the remaining possibilities listed in Table 2.⁶⁷

- (9) a. *Den Hund den ich fütterte* German
 the dog that I fed
 $> \mathbf{B}_x$: [ich]_S/(S\NP) [fütterte]_{(S\NP)\NP}
- b. *John noticed suddenly the man with the big black briefcase.*
 $< \mathbf{B}_x$: [noticed]_{VP/NP} [suddenly]_{VP\VP}
- c. *I offered, and may give, a flower to a policeman.*
 $> \mathbf{B}^2$: [may]_{(S\NP)/VP} [give]_{(VP/PP)/NP}
- d. *Adam dilenci-ye sadaka, kadın çocuğ-a mendil ver-di usul-ca.*
 man beggar-DAT alms woman child-DAT napkin gave gently
 ‘The man gently gave alms to the beggar, and the woman a napkin
 to the child.’ Turkish
 $< \mathbf{B}^2$: [verdi]_{S\NP_{nom}\NP_{dat}\NP_{acc}} [usulca]_{(S\NP_{nom})|(S\NP_{nom})}
- e. *Adam dilenci-ye sadaka, kadın çocuğ-a mendil usul-ca ver-di.*
 $> \mathbf{B}_x^2$: [usulca]_{(S\NP_{nom})|(S\NP_{nom})} [verdi]_{S\NP_{nom}\NP_{dat}\NP_{acc}}
- f. $< \mathbf{B}_x^2$: [showed]_{(S\NP)/NP/NP} [gently]_{(S\NP)\(S\NP)}
- g. *Welke boeken heb je zonder te lezen weggezet?* Dutch
 which books have you without reading away-put
 $> \mathbf{S}_x$: [zonder te lezen]_{(VP/VP)\NP} [weggezet]_{VP\NP}
- h. *He is the man I will persuade every friend of to vote for.*
 $> \mathbf{S}$: [persuade every friend of]_{(VP/VP)/NP} [to vote for]_{VP/NP}
- i. *Welche Artikel hast du abgelegt ohne zu lesen?* German
 which article have you away-put without reading
 $< \mathbf{S}$: [abgelegt]_{VP\NP} [ohne zu lesen]_{(VP\VP)\NP}

j. *What book did you lend without reading and send without understanding to Harry?*

< \mathbf{S}''_{\times} : [lend]_{(VP/PP)/NP} [without reading]_{(VP\VP)/NP}

k. *Kitab-ı Ahmet'e dergi-yi Ayşe'ye oku-ma-dan ver-di-m*
 book-ACC A-DAT mag.-ACC A-ACC read-NEG-ABL give-PERF-1S
 'I gave without reading the book to Ahmet and the magazine to Ayşe.'

Turkish

> \mathbf{S}''_{\times} : [okumadan]_{(VP/VP)\NP_{acc}} [verdim]_{(VP\NP_{dat})\NP_{acc}}

The reader can consult Steedman (1996b, 2000b, 2011), Steedman and Baldrige (2011), Baldrige (2002), Hoffman (1995), Hoyt and Baldrige (2008), Szabolcsi (1992), Jacobson (1990, 1999), Prevost (1995), Komagata (1999), Trechsel (2000), Bozsahin (1998, 2002) and the references cited in these works for a comprehensive list of syntactic constructions studied in detail from this perspective, including, gapping, coordination, relativization, cross-conjunct extraction, control, raising, passives, binding, scope, heavy NP and dative shift, nesting and crossing dependencies, word order and its variation, intonation structure, information structure and word structure. Grammatical organizations that affect a subclass of lexicons en masse, such as accusativity, ergativity and their interaction with subject-, agent- and topic-prominence are upcoming work. McConville (2006), Steedman (2006) provide typological perspectives to CCG.

The discussion in this section gives us a semantically motivated formal base, which we can take to be language invariant. It is the only resource that can constrain a free closure of the lexicon in deriving surface strings, to give us a landscape of possible languages. Possible lexical categories are limited too, as we have seen in Chapter 6 and Chapter 7.

The choices adopted in the remainder of the book among the possible CCG options are as follows. We will assume them in the subsequent chapters where within-school differences are less important than different perspectives on syntax-semantics.

- (i) A freely generating binary **BTS** system, which makes no reference to substantive categories.
- (ii) No freely generating unary rule. Unary rules are lexical rules—after all they do not combine, and they are part of radically lexicalized grammars, hence by definition they must refer to substantive categories.

- (iii) A proposal to include the binary **O** in the system, due to its effects on constituency.
- (iv) No wrap. Therefore, a strictly combinatory system arising from adjacency. Recall that **C** is not surface wrap; it is a combinator and it is lexicalized.
- (v) A linguistic representation of the predicate-argument dependency structures, the PADS, as the key locus of deciding on the lexicalized syntactic types. The constructive work of this choice will be more evident in the next chapter.

The book does not cover matters related to binding and quantifier scope, therefore it can say nothing about LF as a level. Three main proposals are discussed in some detail (Chapter 6). The analyses in the next chapter makes no use of LF as another rule system, or appeal to a system of constraints on binding.

- (vi) No conditions on derivations. All conditions are generalizations and constraints over the syntactic types in the lexicon.
- (vii) Only the basic categories and the slash bear features of relevance to syntax, i.e. morphosyntactic features. Thus only these features are visible to syntax. In effect, this is equivalent to saying that unification does no linguistic work, except to simply match the categories in rule application by term unification (see Pareschi and Steedman 1987 for some discussion). This is in accordance with the agenda of seeing the limits of order doing all the work in syntax and semantics.

Chapter 9

The semantic radar

A syntactocentric view of the landscape of syntactic constructions suggest that they fall into classes because their *syntactic* differences are empirically discernible. Bounded constructions such as passive, reflexive and control are clause-bounded, whereas constructions such as relativization and topicalization are not (and why the clause?). CCG's syntacticization of the combinators as the driving force of the computation of semantic dependencies might suggest that it is likewise syntactocentric in their explanation.

This chapter attempts to show that this assumption would be wrong. The reason has already been implicated in the radical lexicalization of Bach's rule-to-rule hypothesis, so that *codetermination* of syntactic types and semantic types is the key to understanding why constructions manifest themselves the way they do. From this perspective, (un)boundedness must be explained, rather than assumed as some kind of syntactic taxonomy, sometimes with hypergrammatical syntactic principles doing the explaining for their syntactic distribution (e.g., subjacency, the a-over-a principle, different kinds of traces and their governance, exceptions to syntactic projection of expletives, chains, phases, differential linking between the argument structure and dependency structure, etc.). From the perspective of order-caused combinatory syntax and semantics, the explanation lies in the syntax-semantics interaction, and for that we need to see how semantics can shape the syntactic types. The same conclusion seems inescapable for understanding language acquisition and "competence" in competence grammars.

This chapter surveys several domains that force us to bring semantics into play in the explanations. Just how much we must readjust our semantic radar in the grammar might sound like a grandma's recipe for cooking: not too much, not too little. I elaborate in the chapter in more detail. We cannot go as far deep as concepts, and suggest that semantics completely determines syntax, or that syntax could work with semantic types. Nor can we stay with what little information the syntactic types can provide us in lieu of semantics, and suggest that syntax completely determines semantics, or do semantics with syntactic objects.

In all the cases we are going to cover, the semantics that must take part in the process are the individual's *hypotheses* about meanings, i.e. the predicate-

argument structures and dependency structures which must arise from (or feed into) grammars. The construal of these meanings, either by individual experience or by social construction as suggested by Halliday (1978), is the real thing, the experience itself, not a hypothesis. The manifestation of PADS objects in the hypotheses, such as the (e, e) type for pronouns, or (e, t) , $((e, t), t)$, compliment the picture by pinning down their model-theoretic interpretation, but the crucial involvement of the lexical predicate-argument structures will be the decisive factor for syntactic types.

1. Boundedness and unboundedness

There seems to be two ways that lexicalized predicate-argument structures (e.g. verbs) can manifest themselves in syntax, assuming that we are confining ourselves to participant-taking elements, i.e. words with a thematic structure: (i) heed a local argument, or (ii) heed an argument of an argument. From the view of order-instigated semantics, there seems to be no other option.

The first option leads to a theory of voice. Our purpose here is to understand why it is clause-bounded. Their differing possibilities, for example, why the passive targets objects, the reflexive reduces arguments on themselves sparing the subject, and the reciprocal correlates them in the manner of the reflexive, are of course part of the explanation. Steedman's LF, Jacobson's type-shifting rules, and Szabolcsi's constitutive principles of grammar mentioned in Chapter 6 are combinatory attempts at an explanation. Here I will concentrate on (un)boundedness, and use the passive as the first example.

1.1. The passive

It is well-known that the passive cannot cross clause boundaries. (1b) attempts to passivize the embedded predicate of (1a), where the promoted object is not local. (1c) is an attempt to passivize the matrix predicate while promoting the embedded object to subject. (1d) passivizes the matrix predicate where the embedded subject is demoted to a *by*-phrase. This is not a passivization of (1a).

- (1) a. *His closest friend claimed that Kafka loved chemistry.*
- b. **Chemistry claimed that was loved by Kafka.*

- c. **Chemistry was claimed by his closest friend that Kafka loved.*
 d. **That Kafka liked chemistry was claimed by Kafka.*

A purported “long-distance passive” would be misleading, because it would in fact be clause-bounded passivization followed by some other syntactic process. In (2a–b), the process is topicalization by fronting from the embedded clause in brackets. It is not the Turkish equivalent of (1b). It is grammatical because, unlike English, Turkish is a pro-drop language and it allows scrambling to the topic or the postmatrix-verb position from any level of embedding. Example (2c) would be the true long-distance passive where the matrix verb of (2a) is passivized but the matrix subject reduces the embedded predicate.

- (2) a. *Wittgenstein* [*Kafka'nun kimya-yı sev-diğini*]
 W K-3s C-ACC like-COMP
bilmiyor-du.
 not know-PERF
 ‘Wittgenstein did not know that Kafka liked chemistry.’ Turkish
- b. *Kimya-nun, Wittgenstein,* [*Kafka tarafından sev-il-diğini*]
 C-3s W K by-3s like-PASS-COMP
bilmiyor-du.
 not know-PERF
 ‘Wittgenstein did not know that chemistry was loved by Kafka.’
- c. * [*Kimya-nun Wittgenstein tarafından sev-diğini*]
 C-3s W by-3s like-COMP
bil-in-miyor.
 not know-PASS-PERF

We have yet to see examples such as (1b–d) and (2c) to work in any language. Why is that? It is one thing to say that passive is clause-bounded, and build an entire model of syntactic computation with that understanding of domain of locality, and another to explain why it is so. I will sketch an analysis to exemplify the order-induced view of the syntax and semantics of the construction.

The simplest description of a morphologically-marked passive is that a syntactically and semantically transitive verb becomes syntactically intransitive, where the arity reduction causes the participant-type object to show the morphological signs of a subject (Payne 1997). This will do for our purposes, which is not to give a full account of the passive but to explain its clause-boundedness.

Passive is not a universal phenomenon. Washo lacks a passive; see Jacobsen (1979). When attested, it is always lexically headed by a bundle of features which we can call the passive morpheme. Consequently, no one expects a universal passivizer anymore (say a transformation; cf. Chomsky 1957, Bresnan 1978, Bach 1980). This leaves lexical categories to do the explaining for clause-boundedness across languages.

Since passive is voice (it needs participants), it operates on verbal categories in any language, not just on predicational categories. We need something of the type $S\$|NP$ as a domain, rather than $NP\$$ or $S\$$. The notation uses the dollar convention of Steedman.

The category schema of the passive, $S\$|NP$, can be verified in languages where nonverbal predication is possible, including finite (tensed) matrix clauses. Voice is not possible in such cases (*hasta* can be NP/NP but not $S\backslash NP$):

- (3) a. *Annem hasta.* Turkish
 mother.POSS.1s ill
 ‘My mother is ill.’
 b. **Annem hasta-n-di.*
 ill-PASS-PERF
 for ‘My mother has been taken ill.’

It involves an arity reduction of one argument, where the result type must have at least one argument left to show subject properties, because every tensed clause must be fully interpretable. We can revise our domain to involve two or more participants, i.e. $S|NP\$_i|NP$, and range one less, i.e. $S|NP\$_i$, where the common index on the dollar sign means the same member of the lexical generalization is assumed.

For simplicity I am assuming that the type NP can be made a participant-type phrase in a language. The important distinction we use here between the arguments, the participants and the properties does not necessarily need extra degrees of freedom in a type-dependent radically lexicalized theory as it does in for example Construction Grammar. Participation can be achieved in a type-dependent grammar by type-raising all the NP arguments that are onto S . It suffices for our purposes to note that NP/NP would not be a participant-type but NP can be when it is type-raised. For example, $\lambda x.man/x$ denotes a property; the variable x does not have a syntactic correspondent. $\lambda x.sleep/x$, however, denotes a predicate because on the syntactic side it corresponds to an $S\backslash NP$, therefore its x is a participant. When we type-raise an d' to $\lambda P.Pd'$

we can see the narrowing of roles by the lexical syntax-semantics correspondence: if P corresponds to a syntactic argument-taking object such as a verb with $S|NP\$$ type for some ‘|’, then a' is a participant. If not, then it can be something else, perhaps a property. (In other words, participance and argumenthood arise from lexical distinctions rather than some primitives.) One way to impose the participant versus property constraint in a computationally conservative way is to say that NP/NP is not an argument type that is suitable for type-raising.

We can begin to radically lexicalize the skeletal category of the passive, $(S|NP\$_i)|(S|NP\$_i|NP)$, to encode that subject and object are the participatory roles involved. (We cannot assume from this category that it is always the outermost ‘|NP’ of the domain which is the object. In Welsh, a VSO language, that argument is the subject.) Following Steedman and Baldridge (2011), we get the category for the passive morpheme *-en* in English. I will assume coindexed slashes for the present discussion without notational clutter.

$$(4) \textit{pass}' : -en := (S_{\text{en}} \backslash NP\$) \backslash_{\ell} ((S \backslash NP)\$ / NP) : \lambda P \lambda x_n \cdots \lambda x_2 . P x_n \cdots x_2 \textit{one}'$$

where $x_n \cdots x_2 \textit{one}'$ is pointwise match of arity in $(S \backslash NP)\$ / NP$.

The PADS $P x_n \cdots x_2 \textit{one}'$ fully characterizes the active verb’s argument structure with the terms $x_n, \dots, x_2, \textit{one}'$. P can be $\lambda x \lambda y . \textit{adore}' xy$, but not for example $\lambda y . \textit{adore}' \textit{kafka}' y$. This follows from the fact that *-en* applies to lexical items only (the ‘\(\ell\)’ constraint, equivalently, LEX). Examples of applying *-en* are:

$$(5) \text{ a. } \textit{written}' := S_{\text{en}} \backslash NP : \lambda x . \textit{write}' x \textit{one}'$$

$$\text{ b. } \textit{given}' := (S_{\text{en}} \backslash NP) / NP : \lambda x \lambda y . \textit{give}' y x \textit{one}'$$

where *one'* is a nonpro-term, symbolizing syntactic but not semantic arity reduction. Because of type correspondence in the syntax-semantics pairing, *one'* can only correspond to the least oblique (maximally LF-commanding) argument of P , because it applies last.

This PADS and the LEX constraint are not idiosyncrasies of languages like English and Turkish, where the passive morphologically attaches to the verb. It is not a question of morphology but grammar. A periphrastic passive would have a LEX constraint too, to have access to the thematic structure of the passivized predicate. (We shall see in §4 that there are limited other ways to conspire for the lexical constraint to ensure access to relevant parts of the thematic structure, namely the so-called external argument such as in Jaeggli 1986.)

Consider the Welsh *cael* passive as a case in point. For brevity, and in relevance to *one*,¹ I will only consider the short passive, where the *by*-phrase is not present.

- (6) a. *Cafodd Wyn ei rybuddio.*
 Got.3s Wyn his warning
 ‘Wyn was warned.’ Welsh; Awbery (1976: 210)

I repeat Awbery’s description of the passive, which I used earlier to suggest that a pronoun might be required by syntax: “The passive sentence has a sentence-initial inflected form of *cael* (get) of the same tense and aspect as the verb of the active. This is followed by a noun phrase identical to the object of the active. Then comes a pronoun of the same person, number and gender (if it is 3sg) as this noun phrase, and an uninflected form of the verb in the active” Awbery (1976: 47). The pronoun and *cael* are obligatory; Awbery’s data shows that what is dropped if the noun phrase after *cael* is a pronoun is the subject NP, not the possessive pronoun required by the passive:

- (7) *Cawsom (ni) ein rhybuddio gan y ferch.*
 Got.1pl (we) our warning by the girl
 ‘We were warned by the girl.’ Awbery (1976: 48)

Cael takes part in constructions not involving the passive, for example *Cafodd Emyr lyfr* (Got Emyr a book). Awbery assumes that this is the same *cael*, which I will follow.⁶⁸

- (8)

<i>Cafodd</i> got.3s	<i>Emyr</i> E	<i>lyfr</i> a book
$S_{en}/NP/NP_{3s}$	NP	NP
S_{en}/NP		>
S_{en}		
>		

It suggests that the possessive pronoun and *cael* conspire for a passive reading (9).

- (9)

<i>Cafodd</i> got.3s	<i>Wyn</i> W	<i>ein</i> his	<i>rhybuddio</i> warning
$S_{en}/NP/NP_{3s}$	NP	$S \setminus (S_{en}/NP) / \ell(S/NP/NP_{3s})$	$S/NP/NP$
$\lambda x \lambda y. get'yx$	w'	$: \lambda P \lambda Q. (P one')(Q one')$	$\lambda x \lambda y. warn'yx$
S_{en}/NP		$S \setminus (S_{en}/NP)$	
$: \lambda y. get'yw'$		$: \lambda Q \lambda y. warn'y one'(Q one')$	
$S : warn'(get'one'w')one'$			
<			

Notice that, for Welsh, the argument order in the lexical specification of P for ‘*ein*’ is VSO: $\lambda x \lambda y. \text{warn}'yx$. Note also the +LEX constraint on the syntactic type of P although it is not morphologically attached.⁶⁹

From the restriction that the passive applies to lexical verbs, because it requires access to participants therefore to thematic structure, it follows that the substitution environment which one' faces is always of the form (10a) for a passivizable predicate pred' , not (10b), which would be the semantic reflex of (1b–d), because e.g. (10b) would not be an arity reduction of pred' in P but of some x_i . Notice the same (10a) structure of P for English, after *-en* seing the thematic structure and doing all but one last reduction, repeated here as (10c).

- (10) a. $\underbrace{(\lambda x_1. \text{pred}' x_n \cdots x_2 x_1)}_P \text{one}'$
 b. $\lambda x_1. \text{pred}' x_n \cdots (x_i \text{one}') \cdots x_1$
 c. $\text{pass}'(-en) := (S_{\text{en}} \backslash NP\$) \backslash_{\ell} ((S \backslash NP) \$ / NP) : \lambda P \lambda x_n \cdots \lambda x_2. P x_n \cdots x_2 \text{one}'$

One' as a PADS object could not substitute inside the x_n, \dots, x_2 or x_1 , even if pred' were a complement-taking verb such as *claim*, where the complement clause has its own lambda abstractions, for example $\lambda x \lambda y. \text{love}'xy$ in (1).

That is why the passive is bounded. The thematic structure of an argument is opaque to a predicate. Inner lambdas are opaque to *claim* or any complement-taking predicate, therefore nonsubstitutable. This result translates directly to the syntactic types involved.⁷⁰ The construction arises from the interaction of its constraint with the one-at-a-time substitution in syntax and semantics. This property is not a fortunate convenience of lambda calculus; any syntax-semantics connection based on order alone ought to negotiate a similar correspondence.⁷¹

The universal semantics of the passive (that it needs predications of participatory sort, e.g. verbs) explains why it is *clause*-bounded: the types of NPs involved must be functions from participatory types onto S , i.e. type-raised NPs, to be able to distinguish participatory vs. nonparticipatory events. The Turkish distinction $S \backslash NP$ versus NP / NP arises from this aspect (3), where the type NP / NP is not type-raised.

Therefore, the syntactic boundedness of the passive follows from its semantic dependencies and their syntactic reflection: it applies to lexical verbs. However, the LEX constraint involved in this model is a one-way implication. For example, the passive and the reflexive are bounded, and they both arise from the LEX constraint (Steedman and Baldrige 2011). But bound-

edness does not necessarily imply the LEX constraint. Take control, which is bounded, as shown in (11a), but without the LEX constraint (11b).

- (11) a. *I can persuade Mary_i to persuade the wine taster_j to _{-j/*i} try whisky.*
 b. *I want to (seriously challenge)_{-ℓ} (the LEX constraint).*

Radical lexicalization predicts that the LEX constraint cannot be the whole story about boundedness, because some limited degrees of freedom still exist to conspire for boundedness, which are made available when semantics is considered as part of the hypothesis space. Upcoming work attempts to work out the typology of control from a radically lexicalist perspective.

1.2. The relative

Unbounded dependencies follow from similar semantic considerations. Consider relativization, (12).

- (12) *The field which I can safely claim that Kafka could convince Wittgenstein that Russell might like*

The kind of PADS that we see in such dependencies seems not to arise from the predicate-argument structure of a predicate, but from the predicate-argument structure of the arguments of a predicate. Naturally, we expect the syntactic types to reflect the difference faithfully.

For example, in reflexivization and passivization, where, given a predicate, say $\lambda x \lambda y. pred'xy$, they would reduce or equate x or y argument of the predicate $pred'$,¹ hence they can be sensitive to its thematic roles. Unbounded dependencies seem to leave it to the arguments x and y :

- (13) a. *Adam-ın oku-duğunu san-dığ-ım kitap* Turkish
 man-3s read-COMP.3s think-REL-1s book
 'The book which I think the man read'
 b. *kitab-ı oku-duğunu san-dığ-ım adam*
 book-ACC read-COMP.3s think-REL-1s man
 'The man who I think read the book'
 c. *Sen-in kitab-ı oku-duğunu bil-diğini san-dığ-ım*
 You-2s book-ACC read-COMP.3s know-COMP.3s think-REL-1s

adam

man

‘The man who I thought you knew read the book’

The reason I switched to the verb-peripheral language Turkish is to show that when word order constraints are not there, the semantics of these dependencies seem to know no limits as far as the thematic structure of the embedded verb is concerned. Note also that, to the verb *san* above, the argument structure of *oku* is opaque.

The reason that examples such as (13b) can be ungrammatical in a verb-medial language like English—see (14)—is not the unavailability of this semantics because of the opaqueness of thematic roles, but the word order of the language acting as a further constraint on this construction.

- (14) **The philosopher who I can safely claim that Kafka could convince Wittgenstein that would change the world*

All verb-medial and verb-peripheral languages show this asymmetry, barring of course idiosyncratic restrictions (e.g. Inuit only allows ergative NPs to be extracted, although it is verb-peripheral).⁷²

The path to unboundedness follows the arguments-of-the-arguments track, limited only by external factors such as agreement in Latin relative pronouns, and word order constraints. It is thus a conspiracy of semantics *and* syntax, and all that we need to capture this aspect is a type-dependent conception of a category. Unlike the semantics in (10) where *one'* cannot be associated with any x_i because it needs access to the thematic roles of *pred'*, these dependencies must be blind to thematic roles, and the only way they can do this is to associate it necessarily with an x_i . We get the following semantics of relative pronouns as a result of that, which seems cross-linguistically generalizable:

- (15) $relpro' = \lambda P \lambda Q. (\exists x) and'(Px)(Qx)$

Notice that x is not a syntactic variable, and it is not an argument of a predicate whose thematic structure is transparently visible; P and Q are opaque to *relpro'*.

It follows then that the reason why relativization is an unbounded dependency is because P and Q can have their own syntactic lambdas as well so that x can be passed down to them indefinitely. That would in turn require the argument-taking arguments of P , i.e. *think-*, *say-*, *claim-*, *tell-* like verbs. For example, here is the unfolding of the PADS for the bracketed fragment of the string *the philosopher* [*who I claimed that Wittgenstein adored*]:

(16) *who I claimed that Wittgenstein adored* :=

$$\begin{aligned} &\lambda P \lambda Q. (\exists x) \text{and}'(Px)(Qx)(\text{claim}'(\lambda z. \text{adore}'z \text{witt}')i') =_{\beta} \\ &\lambda Q. (\exists x) \text{and}'(\text{claim}'(\lambda z. \text{adore}'z \text{witt}')i'x)(Qx) =_{\beta} \\ &\lambda Q. (\exists x) \text{and}'(\text{claim}'(\text{adore}'x \text{witt}')i')(Qx) \end{aligned}$$

Radically lexicalizing the semantics of this kind spells the following categories for English. Assuming similarly semantically inspired categories for *claim*-like verbs, the transparent syntacticization of the combinators simply reflects these dependencies on syntax. The crucial steps are shown in (17c).

- (17) a. *that* := $(N \setminus N)/(S \setminus NP) : \lambda P \lambda Q. (\exists x) \text{and}'(Px)(Qx)$
 b. *whom* := $(N \setminus N)/(S \setminus NP) : \lambda P \lambda Q. (\exists x) \text{and}'(Px)(Qx)$
 c. *the philosopher*

$$\begin{array}{cccccccc} \textit{whom} & & \textit{I} & & \textit{claimed} & & \textit{that} & & \textit{Wittgenstein} & & \textit{adored} \\ \hline (N \setminus N)/(S \setminus NP) & & S/(S \setminus NP) & & (S \setminus NP)/S' & & S'/S & & S/(S \setminus NP_{3s}) & & (S \setminus NP)/NP \\ & & & & & & & & & & \xrightarrow{\mathbf{B}} \\ & & & & & & & & & & S/NP \\ & & & & & & & & & & \xrightarrow{\mathbf{B}} \\ & & & & & & & & & & S'/NP \\ & & & & & & & & & & \xrightarrow{\mathbf{B}} \\ & & & & & & & & & & (S \setminus NP)/NP \\ & & & & & & & & & & \xrightarrow{\mathbf{B}} \\ & & & & & & & & & & S/NP \\ & & & & & & & & & & \xrightarrow{\mathbf{B}} \\ \hline & & & & & & & & & & (N \setminus N) \\ & & & & & & & & & & \xrightarrow{\mathbf{B}} \end{array}$$

It would be inconsistent to say that *claim* is capable of doing (16) above and has the type $(S \setminus NP)/NP$, rather than $(S \setminus NP)/S'$. The lambda argument of a $'/NP'$ would not be a syntactic lambda (it might be a property, such as $\lambda x. \text{man}'x$, with a semantic lambda), whereas the semantic counterpart of an S' would be expected to have thematic structure. This is captured in the syntacticized **B** without extra assumption; it is not possible to get the **B***claim'**adore'* effect of the third line of (17c) syntactically from $(S \setminus NP)/NP$ and S'/NP ; we need $(S \setminus NP)/S'$ and S'/NP .

It is important to reiterate the universal claim of the type-dependent radical lexicalization about the syntactic processes. It does not claim that the passive is universally bounded and the relative is universally unbounded. It suggests that these behaviors always arise from the transparent projection of rule-to-rule assumptions of a language in its lexicon. Any behavior that seems universal is a manifestation of the self-organizing constraint that a natural grammar would have limited degrees of freedom if it is combinatory, type-dependent and radically lexicalized.

If all languages do something about voice, it is because it seems to arise from the need to have lexical access to thematic structure, which we showed as the LEX constraint. If a class of lexical items specify lexical access to thematic structure, then by definition the thematic structure's opaque parts are not relevant to them, which might give rise to bounded behavior. If a class of predicates allow complements, e.g. say-that, think-that, etc. then unbounded behavior is possible but not necessary.

This way of thinking predicts that when a phrase is only apparently a complement but not a syntactic clause, we cannot expect unbounded behavior. Such morphological ambiguity might arise in morphologically rich languages. Consider (18a), which morphologically seems to include a subordinate clause (18b has the same phonology for the subordinate verb but different semantics; I disambiguated the examples in morphological glosses). As the semantics of relativization from such clauses show in (18c-d) respectively, the first one does not arise from complement semantics; *house* cannot be an argument of the embedded *show* in (18c), precisely because it is not a subordinate clause but a headless relative, i.e. an NP with no thematic structure (equivalently: it has no lexically-specified syntactic lambda).⁷³

- (18) a. *Ahmet Ayşe'nin ev-i göster-diğ-i-ni vur-muş.*
 A A-3s house-ACC show-REL.3s-ACC shot
 'Ahmet shot the one to whom Ayşe showed the house.'
- b. *Ahmet Ayşe'nin ev-i göster-diğ-i-ni bil-iyor.*
 Ahmet Ayşe-3s house-ACC show-COMP.3s-ACC knows
 'Ahmet knows that Ayşe showed the house.'
- c. *Ahmet'in Ayşe'nin göster-diğ-i-ni vur-duğ-u ev*
 Ahmet-3s Ayşe-3s show-REL.3s-ACC shot-REL.3s house
 'The house at which Ahmet shot the one whom Ayşe showed'
- d. *Ahmet'in Ayşe'nin göster-diğ-i-ni bil-diğ-i ev*
 Ahmet-3s Ayşe-3s show-COMP.3s-ACC know-REL.3s house
 'The house which Ahmet knows Ayşe showed'

In summary, if we get the semantics of a construction right, which is to decide whether the thematic structure (local lambdas) or the opaque structure (inner lambdas) is responsible for its dependency, and typologize the syntactic aspects of the words accordingly, as PCTT and the rule-to-rule hypothesis suggest,⁷⁴ then we get the facts of boundedness and unboundedness in syntax as the corollaries of a purely adjacency-based system.

Only additional constraints on lexicalized syntactic types can stop the semantics of the construction from manifesting itself in a language, such as the word order of English eliminating (14), causing the *that-t* effect, Inuit's ergative NP ban on relativization, or the Latin relative pronoun's strictness about the morphological case of the extracted element.

Syntactocentric proposals such as subadjacency, successive cyclicity (of GB) and slash passing (of GPSG) can be thought of as matters to help us pin down the syntactic side of (un)boundedness, but the phenomena and the differences between them do not need extra mechanisms for explanation other than a type-dependent conception of syntactic category based on adjacency, where the semantic side uses lambdas as a way of constructing associations with thematic roles.

2. Recursive thoughts and recursive expressions

Let us have a look at the appeal to extra mechanisms in grammars for the purpose of understanding other aspects of (un)bounded behavior.⁷⁵

Daniel Everett (2005, 2009) has argued that the Amazon language Pirahã stands as a striking counterexample of not having recursion in its grammar because, among other things, it lacks embedding of phrases. This and other gaps in Pirahã grammar and lexicon he attributes to the speakers' cultural choice of insisting on talking about the immediate experiences of interlocutors only. This property, according to Everett, weakens Chomsky's recent claims that syntactic recursion is a necessary human trait distinguishing the language faculty (see Hauser, Chomsky and Fitch 2002).

The key concept in this argument appears to be syntactic embedding. Clearly, Everett could not be claiming that the Pirahã could not entertain recursive semantics as part of their thoughts, such as the semantics of *I like you, I think I like you, I think you think I like you, You think I think you think I like you*, etc., which we might call the immediate-think language of thought, because these can in principle be part of the immediate experience in his account.

A further test for this conclusion can be constructed. Bring for example an English-speaking 10-year-old, who might produce the sentences above, into an exclusively Pirahã-speaking culture. By Everett's account and that of syntactocentrism, which both decide on recursion by the evidence of syntactic recursion, the recursivity of the underlying thoughts in these expressions

is indisputable. In the course of time the child might drop the English-style embedding syntax, and adopt the Pirahã style—assuming Pirahã syntax is indeed nonembedding as Everett claims, see the criticisms by Nevins, Pesetsky and Rodrigues (2009), Pullum and Scholz (2009). This would not change the conclusion that the child had recursive thoughts to begin with, as the syntactic criteria had been observed in the child before.

A reciprocal experiment on hapless children would suggest the same conclusion. Take a Pirahã child to England. Just because a Pirahã born-and-bred child could utter syntactically recursive expressions after enough exposure to English in an exclusively English-speaking community does not necessarily mean the child has learned to think recursively in the new community.

The uniquely human trait of recursion that Chomsky appears to refer to is syntactic recursion, attributed to *narrow syntax* in Hauser, Chomsky and Fitch (2002). The thought experiment provided above shows that no one would doubt the existence of recursive semantics for all humans. We can take it as common ground and look at its consequences.

What exactly is semantic recursion? Surely the immediate-think language concocted above does not require $\mathbf{Y}think'$, which would require both semantic and syntactic recursion. Recall the formulation of \mathbf{Y} using \mathbf{S} and \mathbf{K} in fn. 35, i.e. without syntactic recursion. The \mathbf{K} is the crucial element in that definition for the present discussion. As Craig proved in Curry and Feys (1958), \mathbf{K} cannot be defined by the other combinators discussed so far. Thus we are either left with the syntactic \mathbf{Y} to get \mathbf{Y} effects, or face the empirically fatal \mathbf{K} in syntax, to have syntactic recursion. No data seems to be forthcoming for either theoretical move.

The knowledge of recursion of the kind the word *think* symbolizes simply suggests that people who can entertain *think'*-like thoughts have a knowledge of their language manifesting the understanding of $\lambda x \lambda P. think' Px$, where x is the thinker and P is the thinker, which can be another thing of the same sort, i.e. something onto type t . This knowledge manifests itself in English as $(S \setminus NP) / S' : \lambda P \lambda x. think' Px$. We do not need syntactic recursion for that even if the category were $(S \setminus NP) / S$. Syntactic recursion means a freely-operating \mathbf{Y} in syntax or its functional equivalent, not an argument which is of the same kind as the result.⁷⁶

Theories such as CCG serve to show that the potential infinity of human languages, in the sense of having no upper bound on sentence length or on the number of sentences, does not force us to assume a recursive syntax, as we have so far managed to live without \mathbf{Y} and \mathbf{K} . The language of a CCG gram-

mar is the closure of the **YKI**-less syntax of Table 2 on the lexical assumptions that constitute the CCG grammar. We can assume this property because of radical lexicalization. Nothing moves and nothing is added or deleted by the universal rules. Thus **Y** or **K** cannot appear out of the blue to yield syntactic recursion, unless they are part of the knowledge of some words, i.e. embedded in a lexical category, for which we have seen no evidence so far.

Thus it is assumed from the beginning that a language can be potentially infinite, not because of syntactic recursion but because of closure, that is, from free operation in syntax. Can we entertain the possibility of finite human languages? Yes, by taking a finite closure of Table 2, up to a limit on sentence size, the number of applications of rules or whatever, on a list of lexical assumptions, and proving that the language in question never exceeds that limit. That seems to be extensionally doable, but barring the potential infringement of the future speakers' rights to break that limit, it is intensionally quite problematic.

If we only stick to the number of sentences that have been spoken in a language *up to* a certain time, then any language is vast but finite. Call the set *E*, for example English spoken up to September 4, 2009, and a lexicalized grammar of *E* would be our theory of *that* English.

Would that theory be useful in *understanding* the language manifested in *E*? Certainly. It can help us understand why, in the history of gathering the *E*-expressions, we have never encountered for example a sentence in which three arguments are extracted out of an embedded clause, or why arguments are coindexed indefinitely rather than predicates. We can also wonder why the finite-French set *F* which is locked and sealed at some time appears to have the same properties.⁷⁷

We can also wonder why we never see in *E* the intonational phrasing (*Three mathematicians in*)(*ten prefer corduroy*), while we see an abundance of (*All mathematicians prefer*) (*and some philosophers detest*)(*corduroy*). This is the true nature of linguistic explanation, and it does not need the infinity assumption to be worthy of interest. It certainly would not need syntactic recursion either, for the presumed set *E* is finite.

Thus Hauser, Chomsky and Fitch's (2002) claim that syntactic recursion is indispensable, and Everett's (2005) use of that result at face value—negatively—to conclude that grammars are constrained by cultural aspects and not by universal aspects, are unwarranted. Any grammar reflects a cultural aspect anyway if two or more people happen to agree that, for them, for example $S \setminus NP: \lambda x.sleep'x$ provides the same linguistic recipe of express-

ing *sleep'*-like thoughts in their language. Radical lexicalization predicts that these constraints have no place in universal syntax, and since there is no other locus for formulating these constraints (e.g. phases, spell-outs, cycles, other levels of grammar etc.), they must go in the lexicalized grammar of the language. This makes the cultural aspect of grammar a truism.

We can identify the collective cause of constraints that shape the Pirahã lexicalized grammar as the immediate experience, as Everett (2009) claims. Such a unique source would be of great interest to grammarians, as well as anthropologists and ethnolinguists. The prediction of CCG is that Pirahã *surface syntax* is a closure of that identified grammar on Table 2, not a separate, parallel or parametric mechanism.

In summary, it is not their purported infinity that makes human languages worthy of studying scientifically. It is the limited nature of syntactically manifesting the semantic dependencies. In other words, we seem to be facing a Humean problem in linguistics, not necessarily a Cartesian, Lockean, or von Humboldtian problem. They have assumed *tabula rasa* or the other extreme, and infinity as creativity *par excellence*. The truth seems to lie somewhere in between.

From a cognitive science perspective, we also seem to be facing an old-Platonic, late-Wittgensteinian and Husserlian problem. Knowledge of language can be constructed, as Plato asserted for all kinds of knowledge. But the construction is up for grabs, rather than drawn from a concept repository of the mind. We need the practice of hypothesizing rightly or wrongly about constructions, which requires the true Platonic skepticism *toward* such constructions *after* knowledge is constructed.

False knowledge of words is knowledge if we think it is true by virtue of constructability, and as long as we are prepared to think otherwise when the states of affairs suggest otherwise, as Hume suggested. Recall that, due to radical lexicalization and the combinatory notion of category, knowledge of words is the knowledge of language. Any initial bias, such as that conceived as “universal grammar”, serves to narrow down the search space for the hypotheses about words. It seems to involve a Wittgensteinian play with nature to sort out enumerable meanings from experience, i.e. from personal history, and with kin to share subjective experiences, and with limited access to theories of other minds, as Husserl claimed. Moreover, we cannot assume that other species which are capable of handling *some* semantic dependencies are not able to cope with these things *among themselves* and with nature. The fact that they may not (be able to) communicate these to us is irrelevant.⁷⁸

If any computable semantic dependency were syntacticizable in language, to epitomize human creativity in the infinite capacity of language, we would already have a linguistic theory: the Turing machine, with a memory bounded by some factor depending on the size of the string of words. Somebody has to come forward with some data beyond near context-freeness to make this a forced move, rather than some stylistic or idealistic choice.

A somewhat secondary but not unworthy objection to Everett's (2009) claim that Pirahã falsifies Chomsky's conjecture (that recursion is essential) follows from formal language theory. Hauser, Chomsky and Fitch's (2002) argument in general and Chomsky's early writings in particular (when he had considered the generative capacity of formal grammars a research agenda for linguistics, for example Chomsky and Miller 1963) argue from a *class* of languages. In a class which is considered adequate for natural languages, there must be enough automata-theoretic power to do recursion and context-free dependencies, whether they are attested in every member or not. That is why we try to identify a class of languages with a characteristic automaton. It does not follow that all languages in the same class are equally demanding, so that we might seek recursion in all of them because we have seen it in one (which Everett appears to think Chomsky argued for, which he did not). Take $\{a^{2^n}\}$ and $\{a^{2^n}\}$. Both are in the same class (of recursive languages).

This point is secondary because the main impetus of the objection is that Hauser, Chomsky and Fitch's (2002) argument about the necessity of syntactic recursion in fact shows the necessity of semantic recursion, and the arguments about recursive semantics are quite strong. So are the facts that they *may* be expressed nonrecursively in syntax. Hixkaryana insists on the nonembedding manifestation of recursive thoughts, such as 'He went to Kasawa, because he was wanting to talk with Kaywerye' or 'she was picking it and eating it' (Pullum and Scholz 2009).

The combinators, and through them adjacency, show that having a syntactic type dominating a tree containing that type does not necessitate syntactic recursion. We need evidence for a **YK** syntax or its functional equivalent. No word or constituent seems to involve these combinators.

We must couch a combinatory system of this sort in a set of interfaces so that we can accommodate experiential differences, given the limited nature of syntacticized semantic dependencies.

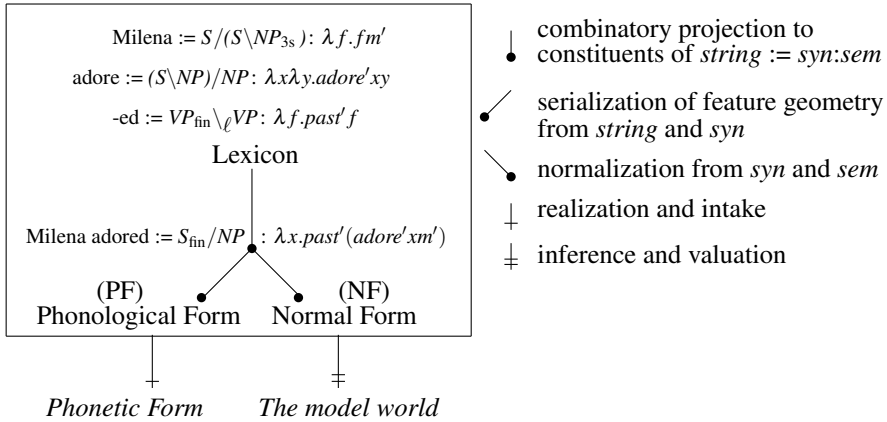


Figure 6. An architecture for linguistic computation.

3. Grammar, lexicon and the interfaces

We need a mechanism to mediate sounds and meanings “out there”, the types in the linguistic system, and multiple experiences. We must keep in mind that the kinds of meanings in question here are hypotheses about what strings mean. They are part of the individual’s grammar. They are not meanings of the sort that makes *The rose saw Kafka*, *colorless green ideas sleep furiously* or *Captain Haddock is the president of the Society of Sober Sailors* to be unacceptable or dubious. This point of clarification cannot be emphasized enough, as Chomsky does quite frequently, for example Chomsky (2000: 199:fn.18) as of lately.

The standardly assumed inverted-Y diagram of linguistic architecture in Figure 6 serves as a good base for adjacency syntax, provided that we put semantics in the frame and out. I use italics outside the box to symbolize that what takes place inside is discretely represented, and what is outside is probably not, e.g. sound and light waves, time- and space-varying images, objects, air pressure etc. The CCG architecture can be thought of as Figure 6 too. Any item in the lexicon that has a syntactic type can take part in the combinatory projection, which is handled by the invariant dependencies of Table 2 without any intermediaries. That is to say that the grammar is radically lexicalized.⁷⁹

As implicated by the direct translation of the combinators' semantic dependencies to their syntacticized counterparts, every constituent gets a syntactic type and an interpretation. The notion of constituency is likewise syntacto-semantic: anything that can be combined by syntacticized combinators is a constituent, including the traditional ones such as *adored Kafka* in *Milena adored Kafka*, and also *Milena adored*. Its constituent behavior is attestable in syntax: *Milena adored* and *I believe Wittgenstein might have liked Kafka*.

The constituent string which carries a syntactic type and an interpretation relates to the phonological form and semantics, which form the linguistic system's gateway to articulatory and intensional-conceptual interfaces. The normal form at the linguistic end of the interface is the PADS normalized on all kinds of conversion, where the applicative structure of the semantics is revealed. Both have perceptual correlates, speaking and for example world- and object-tracking. It is clear that in Figure 6 the mediator of the PF-NF relation is the syntactic type.

The need for PF and NF to communicate with the interfaces to and from arises from semantics as well. Steedman (2000a) has shown that some constituencies in English are unaccounted for unless there is a way to communicate intonational features into syntactic types, and through them to PADS. That is why normalization (and its reverse, abstraction) must heed both the syntactic type and semantics. The model world imagined by the speaker-hearer to which it is anchored outside the linguistic architecture needs no such linguistic mechanisms. We can safely assume that the referents of the PADS terms such as *she* are known to the speaker anyway in a purely applicative form. For example, the referent of *she* was Kafka in the utterance *Kafka wrote Milena many letters; she was adored*, when uttered by me at noon February 1, 2010. These terms are abstractions only to the linguistic systems of the speaker and hearer, which means that the PADS is only one step away from a model-theoretic interpretation.

It seems clear from Steedman's (2000a) work that constituency and intonational phrasing coincide in languages where tunes are at liberty to do syntactic work. (This is not the case in tone languages.) The question is how to decide which is the determinant, and whether it arises from grammar. These issues relate to compositionality. First I note that maximal leftward bracketing allowed by constituency is afforded by CCG. It is not complete left bracketing because of the limited nature of the semantic dependencies, a constraint which seems to be the source of constituency in natural languages.

- (19) a.
- I know that three mathematicians in ten prefer corduroy.*

$$\frac{S/(S\backslash NP) \quad (S\backslash NP)/S'}{S/S'} \xrightarrow{B}$$

- b.
- I know that three mathematicians in ten prefer corduroy.*

$$\frac{S/S' \quad S'/S_{\text{fin}}}{S/S_{\text{fin}}} \xrightarrow{B}$$

- c.
- I know that three math. in ten prefer corduroy.*

$$\frac{S/S_{\text{fin}} \quad (S/(S\backslash NP))/N \quad N \quad (N\backslash N)/NP}{(S/(S\backslash NP))/N} \xrightarrow{B^2}$$

- d.
- I know that three mathematicians in ten prefer corduroy.*

$$\frac{\frac{S/(S\backslash NP))/N \quad N \quad (S\backslash NP)/NP \quad NP}{S/(S\backslash NP)} \xrightarrow{B}}{S/NP} \xrightarrow{B}}{S} \xrightarrow{B}$$

CCG cannot make a nonconstituent interpretable, in semantics or in information structure, thus it makes the narrow claim that constituency is the determinant.

The claim is empirically falsifiable. All legal bracketings are attestable. Take the kind of constituency exemplified in (19c). The prefix up to and including the word *three* can behave as a constituent: *I know that every and you think that some geometers like Euclid.*⁸⁰ The impossible bracketings are the impossible constituents (parentheses show intonational phrasing): **(Three mathematicians in)(ten prefers corduroy)*, as shown in the latter part of (19c).

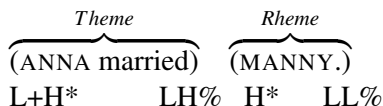
Second I note Steedman's (2000a) observation that, although tunes can lay over different kinds of syntactic constituents, and in different orders, they do the same thing to the phrases on which they are superimposed:

- (20) a.
- Well, what about MANNY? Who married HIM?*
- Steedman

(2000b: 98)

$$\begin{array}{c} \text{Rheme} \qquad \qquad \text{Theme} \\ \underbrace{\text{(ANNA)}}_{\text{H}^* \text{ L}} \quad \underbrace{\text{(married MANNY.)}}_{\text{L}+\text{H}^* \text{ LH}\%} \end{array}$$

- b.
- Well, what about ANNA? Who did SHE marry?*



Pitch accents are designated by H (for high), L (for low) and their combinations. The tone associated with the stressed syllable is designated by suffixing a ‘*’ to the tone. Following the Pierrehumbert and Hirschberg (1990) model of English intonation, we can assume a prosodic organization of intermediate phrases (ι) which are grouped into intonational phrases (ϕ). Intermediate phrase boundaries are designated by L and H, which are distinguished from the intonational phrase boundary tones L% and H%.

Their semantic contribution is crucial to interpretability. Pitch accents on words are reflected in their syntactic types and in their PADS, such as those for *Anna* and *married* above. This process can be assumed to take place presyntactically as suggested by Steedman (2000a), by a rule of associating autosegmental-metrical features with the acoustic correlates of the items in the surface string (or with visual correlates in sign languages). It engenders derivations such as those in Figure 7.

Without this communication with phonology, we cannot assume that H*L is rheme-marking (ρ) and L+H* is theme-marking (θ) in English. This knowledge has its right place in the PADS therefore it must be communicated to it, which can only be done by the syntactic types; see the ‘*’ designations in the derived PADS of strings above, which is used to represent some value of important information. The fact that these are lexical choices (Turkish has no L+H*, and L*H is the theme marker; see Özge and Bozsahin 2010) forces us to assume that the compositional delivery of information structure ought to rely on the lexicalized syntactic types, that is, on a lexicalized grammar.

The delivery of compositional meanings for such kind of constituents depends on the lexical category of the (intermediate) boundary tones. Without their semantics, i.e. theme- or rheme-marking as a side effect on the PADS, the communication from phonology about e.g. stress cannot penetrate the linguistic computation. Many grammatical constituents have been overlooked in linguistics due to this neglect, such as the following:⁸¹

- (21) (PENCERE-Yİ *Ali*), (*kapı-yı*) (MEHMET *kır-dı*) Turkish
 Window-ACC A door-ACC M break-PAST
 ‘Ali broke the window, and Mehmet, the door.’

The example had been rejected on grounds of its claimed oddity in “null context”, but that is precisely the point of bringing in the external factors

<i>Marcel</i>	PROVED L+H*	L-	H%
$S/(S\backslash NP) \xrightarrow{T}$ $\lambda p.p.marcel'$	$(S_\theta \backslash NP_\theta) / NP_\theta$ $\lambda x \lambda y. *prove'xy$	$S\$_i \backslash S\$_\eta$ $\lambda f. \eta'f$	$(S\$_\phi \backslash S\$_\eta) \backslash (S\$_i \backslash S\$_\eta)$ $\lambda f \lambda g. [H](fg)$
S_θ / NP_θ $\lambda x. *prove'xmarcel'$		$S\$_\phi \backslash S\$_\eta$ $\lambda f. [H](\eta'f)$	
S_ϕ / NP_ϕ $\lambda [H](\theta'(\lambda x. *prove'xmarcel'))$			
COMPLETENESS H*	L-	L%	
$S_\rho \backslash (S_\rho / NP_\rho) \xrightarrow{T}$ $\lambda q.q *cmpness'$	$S\$_i \backslash S\$_\eta$ $\lambda f. \eta'f$	$(S\$_\phi \backslash S\$_\eta) \backslash (S\$_i \backslash S\$_\eta)$ $\lambda f \lambda g. [S](fg)$	
		$S\$_\phi \backslash S\$_\eta$ $\lambda f. [S](\eta'f)$	
		$S_\phi \backslash (S_\phi / NP_\phi)$ $\lambda [S](\rho'(\lambda p.p *cmpness'))$	
S_ϕ $\lambda [S](\rho'(\lambda p.p *cmpness'))(\theta'(\lambda x. *prove'xmarcel'))$			
$S_\phi : *prove' *cmpness' marcel'$			

CCG derivation of *Marcel proved completeness*,
in response to *What did Marcel prove?*
adapted from Steedman (2000a: exx.67-68)

Figure 7. CCG and information structure.

into the linguistic system in limited ways, to see the *potential* constituencies demanded by compositional semantics. The example is perfectly grammatical, and the following contextualization proves it. Notice that it is not nonlinguistic recovery from the context or emphatic stress. Note also that the intonational phrases are delivered as semantically interpretable syntactic constituents, which are solely responsible for bringing out their information structure:

- (22) a. *Ben, kapı-yı ALİ kır-dı zanned-iyor-du-m.*
I door-ACC A break-PAST think-IMPF-PAST-1s
'I thought Ali broke the door.'

b. <i>Hayır</i> , (PENCERE-Yİ	L-	<i>Ali</i>)	H-
No	window-ACC	L*	A
	$\frac{S_\rho / (S_\rho \setminus NP_{\rho,acc})}{S_I / (S_I \setminus NP_{I,acc})}$	$\frac{S\$I \setminus S\$_\rho}{(S_\theta \setminus NP_{\theta,acc}) / (S_\theta \setminus NP_{\theta,acc} \setminus NP_{\theta,nom})}$	$\frac{S\$I \setminus S\$_\theta}{(S_I \setminus NP_{I,acc}) / (S_I \setminus NP_{I,acc} \setminus NP_{I,nom})}$
	$\xrightarrow{>T}$	$\xrightarrow{>T}$	$\xrightarrow{>B}$
	$<$	$<$	$<$
	$\xrightarrow{>B}$		
	$S_I / (S_I \setminus NP_{I,acc} \setminus NP_{I,nom})$		
<i>(kapı-yi)</i>	H-	(MEHMET	<i>kır-dı.</i>) L- L%
	L*	H*	M
	door-ACC	break-PAST	
	$\frac{S_I / (S_I \setminus NP_{I,acc})}{S_I / (S_I \setminus NP_{I,acc})}$	$\frac{S_I \setminus NP_{I,acc}}{S_I \setminus NP_{I,acc}}$	$\xrightarrow{>B_x}$

CCG derivation of the constituents in (21):

‘No, Ali broke the window, and Mehmet, the door.’

The example also shows that constituent structure, dependency structure, information structure and functional structure can diverge in various ways, and the simplest way to bring them together is to have them communicate through the syntactic type, rather than devise separate mechanisms for each aspect. The first coordinand above is a nontraditional constituent. The new or important information is spread over the string, and the functional roles of that information are not aligned (window is the object and Mehmet is the subject). Such divergences might suggest multistratal syntax, constraint-ranking in syntax, or “syntax in LF” where we are forced to do some semantic computation in LF using distributional syntactic categories (N, V, A, P) and semantic features in them. No extra mechanism is needed if we have combinatory categories with limited semantic information, which are kept separately but in tight relation to syntactic types.

4. Making CCG’s way through the Dutch impersonal passive

It is not surprising that the most striking empirical challenges to radical lexicalization arise from semantics, in particular from some semantic criterion that can be associated with a class of syntactic objects in seemingly conflicting ways in constructions, such as in unergativity, unaccusativity, and telicity. For example Dutch syntax is known to demand from verbs a particular choice of telicity in auxiliary selection, and another for passivizability (Zaenen 1993). The potential cooccurrence of these constructions makes the problem even more challenging.

It should be clear by now that radical lexicalization as a research program does not mean an easy way out of such problems, such as assuming for Dutch two lexical entries for the same verb, one used for auxiliary selection and the other for passivization. Unless there are compelling empirical reasons to have distinct entries for the verb, most importantly a difference in word meaning, such formal clutter in the lexicon is unacceptable.

I will summarize the problem from the perspective of construction grammar of Goldberg (1995), who follows Zaenen (1991). The impersonal passive requires atelic verbs and verb phrases:

- (23) a. **Er werd opgestegen.* Goldberg (1995: 15)
 'There was taken off.'
 b. *Er werd gelopen.*
 'There was run.'
 c. **?Er werd naar huis gelopen.*
 'There was run home.' Dutch

A class of adverbs apparently related to atelicity can improve judgments:

- (24) a. *Van Schiphol wordt er de hele dag opgestegen.*
 'From Schiphol there is taking off the whole day.'
 b. *Er werd voortdurend naar huis gelopen.*
 'There was constantly run home.' Goldberg (1995: 15)

This aspect seems to contrast with auxiliary selection, which does not change depending on the adverb's atelicity, and insists on the verb's telicity (atelic verbs select *hebben* rather than *zijn* 'is'):

- (25) a. *Hij is opgestegen.* Goldberg (1995: 15)
 'It has taken off.'
 b. *Hij is dagelijks opgestegen.*
 'It has taken off daily.'

Goldberg takes these facts to suggest that the semantics of the impersonal passive cannot depend only on the semantics of the lexical items involved—particularly verbs. The semantics of the construction itself must play the key role. I will sketch a radically lexicalist scenario for the same construction to show that this view may be too pessimistic about the combinatory knowledge of words and what it can do. My goal is not to carry the analysis to a full treatment but to show how radically lexicalist thinking, combined with a

combinatory morphemic lexicon (Bozsahin 2002) and the assumption of structure in words, can provide a solution to the fragment in (23–25).

I will assume for simplicity and ignoring other aspects that the impersonal passive, the unergative verb and the unaccusative have the following lexical syntactic types in Dutch ('atel' is an abbreviation for TELIC=-, and 'tel' for TELIC=+).⁸²

- (26) $-EN := (S_{i,j} \setminus NP) \setminus_{\star} (S_{atel \in i, Ak \in j} \setminus NP)$
 $lop := S_{atel \in i, Ak=atel} \setminus NP$
 $opgesteg := S_{tel \in i, Ak=tel} \setminus NP$
 $naar := (S_{j,tel \in i} \setminus NP) / (S_{Ak \in j} \setminus NP) / NP$
 $dagelijks := (S_{j,atel \in i} \setminus NP) / (S_{Ak \in j} \setminus NP)$

Ak (for Aktionsart) is a complex feature including telicity. The feature without a label, such as S_{tel} , is VP telicity; it arises from the result type of the Dutch VP, i.e. $S \setminus NP$. The lexical choice of adverbs are also shown, where their passing of the verb's Aktionsart is projective (index j), and their syntactic choice of VP telicity (index i) is more liberal. The indices are for ease of exposition; we can think of them as two different features whose value space is that of the feature TELIC. The two-pathway system is implicit in van Hout (2000), where she also talks about the event structure of VPs, not just verbs, and feature checking of telicity by strong case.

It is easy to see how (23a–b) follow from these assumptions. The dubious nature of (23c) can be explained as well. The first derivation below is illicit, and the second derivation goes through. (The projection of Ak is not shown to save space; cf. (26). Note that, in (27a), VP telicity blocks the derivation, not Ak.)

- (27) a. *Er werd* *naar huis* *lop* *-EN*

$$\frac{(S_{tel,Ak} \setminus NP) / (S_{Ak} \setminus NP) \quad S_{atel} \setminus NP \quad (S_i \setminus NP) \setminus_{\star} (S_{atel \in i} \setminus NP)}{S_{tel,Ak=atel} \setminus NP} >$$

$$\frac{}{***}$$
*naar huis gelopen := **
 b. *Er werd* *naar huis* *lop* *-EN*

$$\frac{(S_{tel,Ak} \setminus NP) / (S_{Ak} \setminus NP) \quad S_{atel} \setminus NP \quad (S_i \setminus NP) \setminus_{\star} (S_{atel \in i} \setminus NP)}{gelopen := S_{atel,Ak=atel} \setminus NP} >$$

$$\frac{}{S_{tel,Ak=atel} \setminus NP} >$$

The difference is whether the passive gets phrasal or lexical scope. Notice that we capture the basics of Goldberg's and Zaenen's insight, that the construction itself brings something extra to the example, by letting the adverbial decide the overall telicity rather than the verb, if there is an adverb. Otherwise it is the verb. This seems consistent with the observation that these cases are restricted to a certain class of adverbials, i.e. to certain heads of adverbs (*naar* is telic, *voordurend* atelic, etc.)

The potential derivation for the speakers who marginally allow (23c) depends on the lexical scope for the passive as shown in (27b). The possibility of a phrasal scope however is a forced move in the current state of affairs because of (24), where it is needed for telic verbs as shown in (28–29) (atelic verbs continue to prefer the lexical scope for the passive). The Ak feature is ignored here as it plays no critical role in the derivations.

(28) *Van Schiphol wordt er*
de hele dag opgesteg -EN

$$\frac{\frac{(S_{\text{atel,Ak}} \backslash NP) / (S_{\text{Ak}} \backslash NP) \quad S_{\text{tel}} \backslash NP \quad (S_i \backslash NP) \backslash_{\star} (S_{\text{atel} \in i} \backslash NP)}{S_{\text{atel,Ak=tel}} \backslash NP} \rightarrow}{S_{\text{atel,Ak=tel}} \backslash NP} \leftarrow$$

Once again the adverb decides the telicity because of its syntactic type, which can compose over other adverbs as in the case of (29). This is how the telicity induced by *naar* can be shifted to atelicity by *voordurend* in CCG.

(29) *Er werd voordurend naar huis lop -EN*

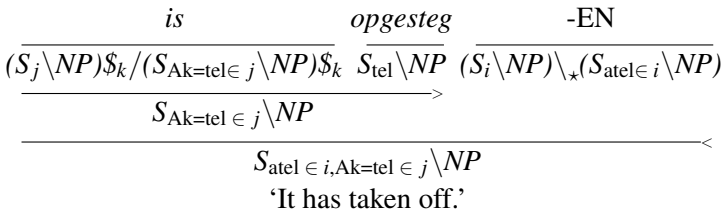
$$\frac{\frac{(S_{\text{atel}} \backslash NP) / (S \backslash NP) \quad (S_{\text{tel}} \backslash NP) / (S \backslash NP) \quad S_{\text{atel}} \backslash NP \quad (S_i \backslash NP) \backslash_{\star} (S_{\text{atel} \in i} \backslash NP)}{(S_{\text{atel}} \backslash NP) / (S \backslash NP)} \rightarrow_{\mathbf{B}}}{\frac{\text{gelopen} := S_{\text{atel}} \backslash NP}{S_{\text{atel}} \backslash NP} \leftarrow} \leftarrow$$

The determinant role of the adverbials by which they take any VP but return telic or atelic VPs depending on their lexical semantics contrasts with auxiliary selection, where the lexical type of the auxiliary selects the verb class, e.g. telic for *zijn* and atelic for *hebben*. It is a domain restriction, e.g. $(S_i \backslash NP) \$k / (S_{\text{Ak=tel} \in i} \backslash NP) \k for *zijn*, which also generalizes over arities.

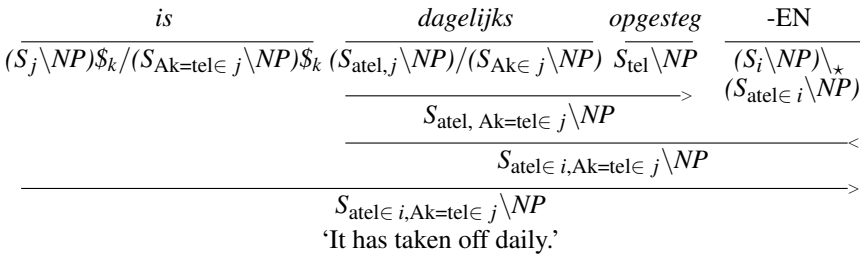
Thus *zijn* and *hebben* look at the Aktionsart (Ak) projected from the verb, whereas the impersonal passive looks at the telicity of the VP with or without adverbial modification. Without an adverb, the telicity of the VP arises from the telicity of the verb. With the adverb, the telicity of the VP is the telicity of

the principal adverb. The Aktionsart of the verb is always projected onto the VP as Ak, without the adverb's intervention, and telicity is projected as a part of it. All these properties are preserved in (30). van Hout (2000) corroborates further for this complex state of affairs which is nevertheless radically lexicalizable, that projecting only the event structure of the verb is not enough.

(30) a. *Hij*

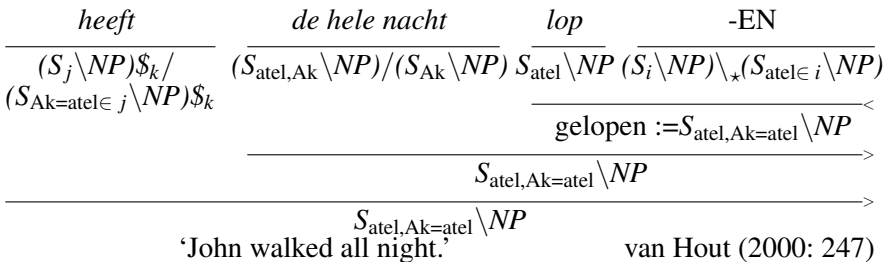


b. *Hij*



Here is the case where the verb is atelic, and chooses the other auxiliary. This is of course descriptively speaking because as the syntactic types show, the auxiliary does the verb-kind selection in the analysis. Notice that the telic adverbs cannot stop the auxiliary from seeing the verb's Aktionsart (Ak) feature (31b). They yield ungrammaticality for independent reasons: the telicity of the VP. Its interaction or lack of it with the verb's Aktionsart is resolved by radical lexicalization.

(31) a. *John*



b. **John*

van Hout (2000: 247)

<i>heeft</i>	<i>in een uur</i>	<i>lop</i>	-EN
$(S_j \backslash NP) \$_k /$ $(S_{Ak=atel \in j} \backslash NP) \$_k$	$(S_{tel, Ak} \backslash NP) / (S_{Ak} \backslash NP)$	$S_{atel} \backslash NP$	$(S_i \backslash NP) \backslash_{\star} (S_{atel \in i} \backslash NP)$
$S_{tel, Ak=atel} \backslash NP / (S_{tel, Ak=atel} \backslash NP)$		$\text{gelopen} := S_{atel, Ak=atel} \backslash NP$	
>B		<	

‘*John walked in an hour.’			

One loose end in this preliminary analysis is of course incorporating Dutch scrambling into it to see its effects on the impersonal passive's scope-taking, which I leave to further study, as Goldberg, van Hout and Zaenen do. With phrasal scope versus lexical scope distinctions, it seems possible to work out a projection scenario where any VP material in CCG's sense is composed over as above for the passive, or lexically chosen by it. The phrasal option for the passive is not a far-fetched theoretical option either; it is the only possibility in Welsh, which has a periphrastic passive (§1), and no morphological marking on the verb.

In summary, the auxiliary is the head of auxiliary selection, and the adverbial is the head of VP telicity if present, otherwise it is the verb, and the verb's telicity always projects. All of these follow from the uniquely lexicalizable syntactic and semantic assumptions about the category of heads in Dutch. Notice also that the assumptions of §1 about the passive, that it needs to see the thematic structure of the verb, which translates on the syntactic side to the LEX constraint on the slash as ' \backslash_{ℓ} ' or ' \backslash_{ℓ} ', is still adhered to in the category of -EN (26) in an indirect way. Its syntactic type is not $\$$ -schematized, therefore it must take a one-argument predicate, whose thematic role is therefore visible. This seems consistent with Jaeggli's (1986) insight that passive is an external argument absorber. That argument in our Dutch grammar fragment is the syntactic subject of the unergative or unaccusative verb, due to the $S \backslash NP$ domain for -EN. It must face a verb because of the ' \backslash_{\star} ' constraint, which prevents it from undergoing composition with adjunct NPs and verbs in serial verb constructions. Thus all syntactic work is done by the syntactic types, rather than morphological types and syntactic types such as in Jaeggli (1986).

We would expect the categories S_{tel} and S_{atel} to arise from lexical semantics, as these are associated with words (*naar*, *voordurend*, *gelopen*, *opgesten* etc.), and projected onto syntax from them. For telicity to do the syntactic work, such features must be reflected on the syntactic types. Just how much is projected (and how) is a lexical choice, as predicted by the principle of lex-

ical head government (PLHC), such as the verb's Aktionsart and the VP telicity going their separate ways in Dutch because it is demanded by syntax.⁸³ In our case, feature percolation can happen if these conceptual-semantic features were made part of the feature space of the semantic objects in a lexical PADS, which in turn codetermines the syntactic type. We can presume that this process might take place as qualia (Pustejovsky 1991) or Jackendoff (1997)-style lexical dependency structures. The crucial aspect for the present concerns is that this is quite a limited interface with conceptual structure, to ultimately find its way to the syntactic type, and it can only happen at the lexical level since there is no other level.

Steedman and Baldridge (2011) show that another Construction Grammar favorite, the *way* construction (Goldberg 1995), is similarly radically lexicalizable without any need for extra semantics or syntax over and above lexical items. The construction is headed by the reflexive *his way* (or *her way* etc.):

- (32) a. *Harry slept his way through the final exam.*
 b. **Harry slept Barry's/her/their way through the final exam.*

They provide a lexical semantics and a syntactic type for it, which I repeat below. The participants and their semantics are clear: a lexical verb, a spatiotemporal property and a subject.

- (33) *-his way* := $((S \setminus NP_{3s}) / PP_{loc}) \setminus \ell (S \setminus NP_{3s})$
 $:\lambda P \lambda Q \lambda y. cause'(iterate'(Py))(result'(Qy))$

Radical lexicalization and CCG's transparent projection give us narrow opportunities to make predictions and to check our lexical assumptions about cases where the constructions interact, because nothing can intervene or alter the projection of features and types onto surface syntax, hence we do not need to worry about the degrees of freedom that might be exploited in some linking rule or pre- versus postspellout. For example, we can test the lexicalized reflexive constraint above (the ' $\setminus \ell$ ' type; note the affix assumption in '*-his way*', which is the main input to the narrowed slash). Fronting and node-raising seems unacceptable:

- (34) a. **His way Harry slept through the final exam.*
 b. *Harry_i slept and Barry_j worked his_{j/*i} way through the final exam.*

Thus we do not need assumptions over and above the lexical items and constitutive principles of the lexicon (PCTT, PLHG, etc.) to understand the constructions. Construction Grammar's use of argument roles for constructions,

in addition to the participant roles of verbs to explain the phenomenon, forces one more linking theory into a theory based on mapping principles. Most linking theories leak, as the gradual transition of LFG, the most worked-out linking theory, to optimality-theoretic syntax has shown.

5. Computationalism and language acquisition

Adjacency as the sole basis of all hypotheses about the grammar suggests a computationalist scenario for language acquisition. Here also the kind of semantics we need is quite shallow, and originally distinct from syntactic representation.

First a point of clarification about the book's perspective on cognitive science. The term *computationalism* is yet another source of confusion in cognitive science. There are computational models which are not computationalist, and noncomputerized models which are computationalist. Computationalism suggests that the aspects that make a problem computationally easy or difficult, such as nondeterminism, automata-theoretic resource management, and algorithmic space and time complexity, are significant factors in for example the child's elimination of her hypothesis space in language acquisition. Efficiency of course cannot be the whole story in this endeavor; it will cause tension with expressivity as the child grows, and this aspect has to be part of a model too.

The point can be clarified with an example. Suppose that we are trying to see the role of homonymy and synonymy in communication. We can start with some cognitivist primitives, such as "avoid homonymy" or "disprefer synonymy" to model efficient communication. Or we can show through a computationalist model that in a group of communicating agents having too many homonyms and synonyms cause late convergence to a common vocabulary. Such experiments have been conducted by Smith (2003), De Beule, De Vylder and Belpaeme (2006), Eryılmaz and Bozsahin (2012). The complexity of the task and complexity of life seem to conspire to constrain the behavior, rather than cognitivist assumptions.

There is another interpretation of computationalism in cognitive science and psychology, where it is taken as the agenda of treating symbols as relating to the nature of representations, that is, to their encoding in the mind (see e.g. Bickhard 1996). Computationalism in the broader sense does not need this assumption because computationalist models—whether implemented in

a computer or not—are hypotheses about what connects representations to solutions, not how they are internalized. This is true of connectionism as well, a field which is unfairly left out of computationalism in wholesale by some psychologists. Take for example Elman's (1990) modeling of time, in which a change of input encoding does reflect on the nature of the problem, yet solutions live or die by computational properties. Thus there is no conflict in adopting computationalism as a whole, in addition to interactionism Bickhard has been advocating.⁸⁴

Let us look at some alternatives to computationalism, for example a cognitivist treatment of acquisition. It has been argued that nouns are acquired first (Gentner 1982). That would be a conceptual bias toward names, objects and their perception, hence their first appearance in child language.

Table 3. Tad's first words (Gentner 1982) (AmE).

Age (m.)					
11	dog	16	eye	19	down
12	duck	18	cow		boo
13	daddy		bath		bottle
	yuk		hot		up
	mama		cup		hi
	teh (teddy bear)		truck		spoon
	car	19	kitty		bye
14	dipe (diaper)		pee pee		bowl
	toot toot (horn)		happy		uh oh
	owl		oops		towel
15	keys		juice		apple
	cheese		TV		teeth

For example, Table 3 shows Tad's first words starting at 11 months. They seem to be adult nouns, and whether they are child nouns strictly we have so far no way of knowing. For example, *keys* might also mean *open*, or *dipe*, *clean*. Keren's first words appear to be similarly reinterpretable (Table 4).

20-22 month-old Mandarin children seem to show no noun-verb bias (Tardif 1996). This result and a reinterpretation of the results above might suggest a computationalist perspective, first proposed for machine learning by Zettlemyer and Collins (2005), and adopted for language acquisition by Steedman and Hockenmaier (2007), Çöltekin and Bozsahin (2007).

Table 4. Keren's first words (Dromi 1987) (Hebrew, Israel).

Age m(d)	Child's word	conven. form	
10(12)	haw	(?)	a dog's bark
11(16)	?aba	(aba)	Father
11(17)	?imaima	(?)	
11(18)	ham	(?)	said while eating
12(3)	mu	(?)	a cow's moo
12(3)	?ia	(?)	a donkey's bray
12(8)	pil	(pil)	an elephant
12(11)	buba	(buba)	a doll
12(13)	pipi	(pipi)	urine
12(16)	hita	(?)	going out for a walk
12(18)	tiktak	(?)	sound of clock
12(19)	cifcif	(?)	bird's tweet
12(20)	hupa	(?)	accom. making sudden contact w/ground
12(23)	dio	(dio)	giddi up
12(25)	hine	(hine)	here
12(25)	?ein	(?ein)	all gone
12(25)	na?al	(na?al)	a shoe
12(25)	myau	(?)	a cat's meow

If we take the problem of language acquisition as manifesting a continuous problem space for words and phrases, and if we assume that the hidden variable in the task is the syntactic category to be learned, whereas the observables are a phonological form and the model world, crucially not PADS or a logical form, then we would expect the child to start off with some prior probabilities on invariants of combination, and proceed as she manages to combine rightly or wrongly what she hears as syntactic categories to pair with predicate-argument structures.

For example, upon hearing *eat your veggies*, the child might think *eat* means *eat'*, *veg'* or even *dog'* if there is a dog around when the sentence was uttered. Limited possibilities of combination in CCG, and a conservative understanding of tracking the world (e.g. Siskind 1995, 1996), will sieve most of the wrong assumptions as the child experiences more episodes with eating, dogs and vegetables, eliminating e.g. the hypotheses $N: eat'$ and $S \setminus NP: dog'$.

An algorithm is provided for this task by Steedman and Hockenmaier (2007). My running example of dogs, eating and veggies is fashioned after theirs. The setup is common to all CCG learners, which dates back to Gold's (1967) text model: start with an empty lexicon. For each experience, generate some hypotheses that lead to its successful parse, and update the lexicon. Repeat with the new lexicon. In retrospect, the lexicon will have covered all the strings the learner has experienced, where "something more" in the Humean sense is also learned to cover things beyond a token of experience: the syntactic type as the hypothesis.

It is crucial that what is learned is a syntactic type. In a way it symbolizes the transfer of experience-specific knowledge to reusable knowledge, or perhaps impressions to ideas, to use a more familiar Hume terminology. For example, we can conceive that the passive is learned by exposure, but once learned, it applies to all argument-taking objects of the right sort because acquiring the passive means obtaining a syntactic type for it, which is relevant to verbs of similar type.

The working principle here is that the CCG learner collects personal historical information about derivations of strings—i.e. rule and word use—in the parse-to-learn paradigm, either by adjusting the model parameters (log-linear models), or by updating its trust on categories (Bayesian models), in the manner described by Zettlemoyer and Collins (2005), Steedman and Hockenmaier (2007), Çöltekin and Bozsahin (2007), Clark and Curran (2007).⁸⁵

That is, its task is to estimate $P(c|e)$, either by discriminative (log-linear) models or generative models, where c is a syntactic type and e is the evidence for it in the form of (PF, PADS) pairs, calculated for example by Bayes's rule:

$$(35) P(c|e) = \frac{P(c)P(e|c)}{P(e)}$$

The prior probability $P(c)$ is selected by the learner's history in what she perceives or (rightly or wrongly) understands; it is her current lexicon's syntactic distribution. This is not only constrained by experience; universal constraints filter out some impossible configurations as well.

The Bayesian model sketched so far is not incremental. To estimate the conditional probability $P(E:=e | C:=c)$, we need to find out which parses using the current lexicon and the newly introduced hypotheses give us c , and among them the probability $P(E:=e)$. Some of the earlier experiences will be related to c as well, hence the need to reparse them to get $P(C:=c)$.

Even if we assume that each experience is unique, its subparts are most likely not all unique (otherwise learning would be very hard if not impossible), therefore subparts of e and several c 's must be considered for each experience. For example, eat' veg' might be a new experience when eat' and veg' are not, such as encountering *don't eat all the cookies* and *I like veggies* before. Zettlemoyer and Collins (2005) use a limited category inventory in lieu of universal grammar to constrain the possibilities of new categories for the new experience, and Steedman and Hockenmaier (2007), Çöltekin and Bozsahin (2007) rely on universal principles such as those in §5.2 and Chapter 7.

We need an iterative method which parses the current experience only with the help of the current lexicon—the grammar—and the new hypotheses. For example, we can take a weight w to be the learner's belief that her hypothesis about a certain category is correct. The following oversimplified formula from Çöltekin and Bozsahin (2007) is one example of hypothesis revision. (Log-linear models such as that of Zettlemoyer and Collins 2007 use easily discernible features of parse trees, e.g. number of lexical entries and number of applications of a rule, which takes into account the current lexicon and rule use.)

$$(36) \quad w = w_0(1 + \alpha\beta(1 - w_0))$$

w_0 is the probability (or weight) of the lexical hypothesis c before seeing the input e . If the hypothesis is already in the lexicon, w_0 is the weight of the hypothesis in the current lexicon, otherwise an arbitrary initial value is assigned. New hypotheses can be added although substrings of the current experience have already been seen. For example, if the child thinks $eat := NP:veg'$ and $veggies := S \setminus NP:eat'$ somehow, and the new experience is *no veggies*, we can produce $no := S/NP:no'$ and $veggies := NP:eat,'$ meaning 'no eating'.

The constant α in (36) is the learning rate, which must be part of an experimenter's toolbox. We can assume for the child that it improves with experience. The β in the formula is the learner's new evidence that the category c might help to understand the new experiences. It is calculated as the number of parses of e in which hypothesis c is used, divided by the total number of parses of the experience e .⁸⁶ It gives new support for the category c provided by e . The higher the number of parses that the hypothesis supports, the higher the support value will be. If the hypothesis is used by all the possible parses of the input, the value is 1. The value gets smaller due to the parses that do not include the hypothesis. The final term in the formula, $1 - w_0$, normalizes

the result so that the new weight is in the range (0,1]. The final weight is increased with a value directly proportional to the new trust on c , as shown in (36).

This is inspired by Bayesian hypothesis revision but it is not strictly Bayesian. Firstly, the implicit assumption is that there is no negative evidence, as the probabilities do not decrease. One can see no increase in the weight of a hypothesis as less belief in it, compared to its alternatives whose weight increases. The problem can be alleviated if we can fit a distribution for $P(e)$ in (35), but this is rather difficult if not impossible.

Secondly, the model has no grounds to distinguish infrequent but correct hypotheses from incorrect but frequent ones. In the first case, the belief in a hypothesis would not increase much, and in the second case, it will continue to increase, albeit slowly.

This weakness is required empirically, because the child is assumed to operate in what Gold (1967) called the “text” model, where there is no decider for any experience e whether a hypothesis about it is right or wrong. (A rationalist model for example could take this as a sign that the functional categories are innate, because their overt manifestation is infrequent in early child speech.) From an empiricist perspective, especially with the narrow understanding of computationalism adhered to in this work, incorrect but frequent hypotheses (categories) are bonafide members of the lexicalized grammar of the child, and infrequent but correct hypotheses need more time to materialize in a parse-to-learn paradigm.

The computationalist twist in such models is that only contiguous substrings (including the substrings of words discussed in Çöltekin and Bozsahin 2007) are allowed to bear types, therefore to carry a meaning, and short strings are considered more feasible because the algorithms must consider all such possible pairs, i.e. the powerset of possible PF-PADS mappings, so that we can be sure the child in the end can potentially manage to bring the correct pairing to the fore through experience. Such algorithms will show a bias toward frequent, short or unambiguous strings because these aspects can be shown to ease the task computationally. For example, the powerset construction is exponential on the size of the set, which is the set of hypotheses. Only small values are feasible in a learning model, and the contiguity assumption is a simple way of reducing it from $O(2^n)$ to $O(n^2)$. I repeat Garey and Johnson’s (1979) numbers for differences in growth rates of functions as Table 5 (each unit operation is assumed to take one microsecond). Any n greater than 5 can tell us how these reductions in problem size can play a role.

Table 5. Growth rates of some polynomial and exponential functions, from Garey and Johnson (1979: Fig.1.2)

Time complexity function	size n					
	10	20	30	40	50	60
n	.00001 second	.00002 second	.00003 second	.00004 second	.00005 second	.00006 second
n^2	.0001 second	.0004 second	.0009 second	.0016 second	.0025 second	.0036 second
n^3	.001 second	.008 second	.027 second	.064 second	.125 second	.216 second
n^5	.1 second	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	13.0 minutes
2^n	.001 second	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 second	58 minutes	6.5 years	3855 centuries	2×10^8 centuries	1.3×10^{13} centuries

The computationalist model is falsifiable. The computationalist assumptions would be wrong if we can show that the length of the strings, their ambiguity and their frequency do not play a key role. For example, a nouns-first cognitivist theory can show one of the following to refute the computationalist assumptions: (a) some short verbs are not learned early even when they are frequent and unambiguous, (b) some frequently-used long nouns can be learned early, (c) infrequent but short nouns can be learned early, and (d) some ambiguous but short nouns can be learned early. In all these cases, some strong computationalist assumption would be at risk.

The computationalist view suggests that we take another look at the results. For example, for both Tad and Keren, long words seem to be rhythmic repetitions, i.e. they engender no ambiguity as the string becomes longer. Short nouns can be child verbs too.

Early acquisition of verbs seems possible (Brown 1998). Interestingly, the verbs that Tzeltal children acquire early seem to be argument-specific therefore less ambiguous than opaque verbs. For example, eating tortillas, eating beans and eating in general (as in a question) are different words in Tzeltal.

Some early-acquired verbs such as those for go, make, come are not argument-specific, but they are the most frequent verbs in the language.

Brown is not suggesting a verbs-first alternative to the nouns-first proposal based on these findings. She shows that the amount of nouns and verbs produced from the early one-word stage and prevocabulary explosion are more or less the same. This is what we would expect when verbs are specific and/or frequent, and nouns and verbs are equally rich in morphology, as in Tzetal.

Computationalist models are possible only if we start with the assumption that the child has access to some semantics, not just to meanings out there but to some hypothesis about what she thinks they mean, that is, an access to a PADS.⁸⁷ The environment and what she hears from it might be related to that semantics because her attention is directed by adults when she is spoken to. Evaluating the hypotheses of PF-PADS pairs is feasible if we assume adjacency. With empty categories or with syntactic assumptions on the child's understanding (e.g. S, VP etc.) rather than semantic ones, the number of hypotheses to consider would be prohibitive. One such proposal, which seems only apparently congenial to computationalism, is Hawkins's (1994) processing-based account of establishing the basic word orders in languages. In his model, as well as in Kayne's (1994) where movement and empty categories are bound to come up for consideration at every step of processing, the number of possibilities for a parser to consider in the parse-to-learn paradigm is quite unconstrained.

6. Stumbling on to knowledge of words

The process described in the previous section gives us a recipe to devise explicit tokens of knowledge representation for the child's potential hypotheses about the words. Their statistical nature might raise doubts about whether this way of thinking can live up to the task of explaining why one-word and two-word stages of children, and the vocabulary explosion that follows soon afterwards, more or less appear around the same time for most children. The first thing to note about this doubt is that no-one claims children start *tabula rasa*; the task-specific knowledge, namely the lexicalized syntactic type, must have severe constraints on its distribution. This is the task of CCG as a linguistic theory, in lieu of a biologically determined universal grammar in generativism. Secondly, now that we can radically lexicalize all the rules of any natural grammar, that is, we have only the knowledge of words to work

with in hypothesizing, we must show what the experience can do to the rules in Shimon Edelman's sense, and how. In such experiments we are reminded of the opening words in his personal web site: "rationalists do it by the rules, empiricists do it to the rules." In a radically lexicalized combinatory grammar, a word's category *is* the grammar rule because it is an intensional recipe.

This section presents a thought experiment about how a fairly intuitive notion of *word* as a grammatical-historical object can be read off from the lexicon. Radical lexicalization and the experiential-semantic understanding of "standing on its own in a string" appear to be sufficient for this process. The experiment is inspired by computational language learning in the manner of Zettlemoyer and Collins (2005), Steedman and Hockenmaier (2007), which are inspired by cross-situational learning of Siskind (1995, 1996) and CCG, which led to similarly inspired computational models of learning string-meaning correspondences (e.g. Villavicencio 2002, Bos et al. 2004, Steedman 2005a, Fazly, Alishahi and Stevenson 2010, Kwiatkowski et al. 2010, 2011), all of which go back in spirit to late-Wittgenstein (1942), Quine (1960) and Gibson (1966).

The difference of the present experiment from these works is that they presume the notion of word and suggest a model of how their meanings may arise from use. I will try to suggest a thought experiment about how words may arise in the first place. My starting point is to assume that children can detect patterns in phonological strings. We can take these patterns to be child morphemes, but we need not start with the morpheme. In a related study, Çöltekin and Bozsahin (2007) showed that if we start with syllables (i.e. if only syllables are assumed to be discernible by the child), and run a scenario similar to Zettlemoyer and Collins (2005) on the Turkish fragment of the CHILDES database (McWhinnie 2000), we get 71% of the emerging lexical items (including bound forms) coincide with that of a model which starts with morphemes, in 24,000 nouns, out of which 56% are inflected. Their syllable model does not make assumptions about root/stemhood, hence we can expect more alignments if we incorporate some prosodic cues about uninflected words, which comprise 44% of the database (Jusczyk, Hohne and Newsome 1999, Thiessen and Saffran 2003 suggest that these cues are at work at very early stages). This is not a bad start to give rise to meanings of things smaller than words.

Consider the word *veggies*. One criterion of Di Sciullo and Williams (1987) for wordhood in the currently discussed sense is that words are more generic than phrases. We have no reason to assume that at the first hearing

of this word it would be generic to the child. Assume that the child has gone through a Quinean series of hypothesis forming where many hypotheses (most of which might be wrong) have been entertained, much like in Siskind (1996).⁸⁸

For example, we can assume that the experience (37) might produce the correct hypotheses in (38a/a'), as well as those in (38b–c), which are the situations in which the string *eat* is not understood as the verb, but the overall experience still spells some kind of predication, simply indicated here by the overall result of *S*. (38d) is another potential set, in which *eat*'s category is correct, but *veggie* and *-s* are off the mark. We can take (38) to be delivered by a parsed-to-learn paradigm of acquisition.

(37) *Eat veggies.*

- (38) a. $eat:=S/NP:eat'$ $veggies:=NP:veg'$
 a'. $eat:=S/NP:eat'$ $veggie:=NP:veg'$ $-s :=NP\NP:plu'$
 b. $eat:=NP:eat'$ $veggies:=S\NP:\lambda x.veg'x$
 c. $eat:=NP:veg'$ $veggies:=S\NP:\lambda x.eat'x$
 d. $eat:=S/NP:eat'$ $veggie:=NP/NP:plu'$ $-s :=NP:veg'$

This experience cannot lead to the hypotheses in (39a–c) because no combinator in syntax can combine them to produce a rightly or wrongly interpretable experience. The distribution of syntactic types *S*, *NP/NP*, *S/NP* etc. are therefore most likely skewed.

- (39) a. $*eat:=NP:eat'$ $veggies:=S/NP:veg'$
 b. $*eat:=S\NP:eat'$ $veggies:=NP:veg'$
 c. $*eat:=S\NP:eat'$ $veggie:=NP:veg'$ $-s :=NP\NP:plu'$

Note also that a predicate-argument structure is part of the child's hypothesis space; it is not the extensional world. For brevity I denoted it with primes. We do not start with the assumption that the child knows *veggies* are *veggies*, where the only unknown would be whether they are Ns or Vs in syntax. Both are acquired.

Now consider a second experience, say (40).

(40) *No veggies.*

This will create more hypotheses about *veggies*. Let us also take into account the nonlinguistic surrounding in the manner of Siskind (1995, 1996), and assume that there is a chocolate bar around when this sentence is uttered. The child might think that *veggies* can mean negation (because of *no*), or that

it could mean chocolate, veggies, or eating (the last one comes from the previous experience). We must also allow for the possibility that she might think “veggies” could mean the noun *veggies*, or that it could be a verb. Hence assuming veggies are veggies would be an oversimplification; both syntactic options must be entertained even if we assume that she has got the string-content correspondence right.

Even in this circumscribed world of two experiences only, the child is exponentially less likely to believe that veggies could mean negation, eating, plural or chocolate, rather than veggies. The sum of 43 hypotheses is calculated as follows.⁸⁹

(41)

Experience 1 (Eat veggies)				
eat :=S/NP:eat'	veggies :=S\NP:veg'	veggie :=NP	:veg'	-s :=NP\NP:plu'
:veg'	:eat'	NP/NP:plu'		NP :veg'
NP :eat'	:plu'veg'		:veg'	
:veg'	:plu'eat'			
	NP :veg'			
	:eat'			
	:plu'veg'			
	:plu'eat'			
Experience 2 (No veggies; with chocolate)				
no :=S/NP:no'	veggies :=S\NP:no'	veggie :=NP	:no'	-s :=NP\NP:plu'
:veg'	:veg'	:veg'		NP :veg'
:choc'	:choc'	:choc'		:choc'
	:eat'	NP/NP:plu'		
	:plu'veg'	:veg'		
	:plu'choc'	:choc'		
	:plu'no'			
	NP :veg'			
	:eat'			
	:no'			
	:choc'			
	:plu'veg'			
	:plu'choc'			
	:plu'no'			

$\frac{14}{43}$ percent of the possibilities, out of a total of 43 chosen above, can relate the string *veggies* to *veg'* as a noun or verb. In contrast, the likelihood of *no* meaning *veg'* is $\frac{1}{43}$, the plural $\frac{2}{43}$. If we keep a local statistic rather than a global one, there would be a set of 36 hypotheses about the set of forms {*veggie*, *-s*}, and $\frac{14}{36}$ percent of it would relate them to *veg'*. The total percentage of associations where the string *veggies* does not include *veg'* is $\frac{22}{36}$. That seems high, but it covers four meanings (plural, negation, eat and chocolate) and four types, which are $S\backslash NP$, NP , NP/NP and $NP\backslash NP$. By Siskind's (1996) *cross-situational inference*, and by CCG's fully lexicalized syntactic

types, the likelihood of *veggies* covering one of these type-meaning correspondences is severely less than the *veggies* := *veg'* connection. I ignore here how the plural can come to be associated with *veg'* using these assumptions in parsing. For example *veggies* can be parsed from *veggie* := *NP/NP:plu'* and *-s* := *NP:veg'*, where both hypotheses are wrong but they yield the intended interpretation *veggies* := *NP:plu' veg'*; see Steedman and Hockenmaier (2007), Zettlemoyer and Collins (2005).

Let us add another experience, (42).

(42) *Veggies gone.*

Before this experience, $\frac{8}{14}$ percent of the *veggies* := *veg'* hypotheses considered this relation to be mediated by *NP*, $\frac{4}{14}$ by *S\NP*, and $\frac{2}{14}$ by *NP/NP*. The new experience can bring in the hypotheses in (43) (for simplicity I assume no other factors).

(43)	<i>veggies</i> := <i>S/NP:veg'</i>	<i>gone</i> := <i>S\NP:veg'</i>	<i>veggie</i> := <i>NP</i>	<i>-s</i> := <i>NP\NP:plu'</i>
	: <i>gone'</i>	: <i>gone'</i>	: <i>veg'</i>	<i>NP</i> : <i>veg'</i>
	: <i>eat'</i>	<i>NP</i> : <i>veg'</i>	<i>NP/NP:veg'</i>	<i>S/NP</i> : <i>gone'</i>
	: <i>no'</i>	: <i>gone'</i>	: <i>plu'</i>	: <i>veg'</i>
	: <i>plu' veg'</i>			: <i>plu'</i>
	<i>NP</i> : <i>veg'</i>			
	: <i>gone'</i>			
	: <i>eat'</i>			
	: <i>no'</i>			
	: <i>plu' gone'</i>			

This time we fortuitously help the child to discern the noun versus verb hypotheses of *veggies*, but we make *plu'* slightly more susceptible because it has more opportunities for combination to the left and right. (To be sure, there are more hypotheses in this three-scene experience, and some of the hypotheses considered are not hypotheses in the parsimonious model of Siskind; my purpose here is to construe a baseline case by making things bad enough for the experiment.)

With the addition of seven more {*veggies*, *veggie*, *-s*} := *veg'* hypotheses to the previous 14, the child is $\frac{11}{21}$ likely to believe the connection is mediated by *NP*, $\frac{4}{21}$ by *S\NP*, $\frac{3}{21}$ by *S/NP*, and $\frac{3}{21}$ by *NP/NP*, in just three scenes. We can assume that a language model, in the sense the term is used in computational linguistics, i.e. as a model to pick some product of probabilities in a parse-to-learn paradigm, will favor the type with higher probability as the primary representative of the word in grammar.

The *NP* hypothesis for the word *veggies* is the top contender after these three experiences, with a total frequency of $\frac{27}{55}$, in which the correct relations,

$$\{\text{veggies, veggie}\} := \left\{ \begin{array}{lll} S \backslash NP: \text{veg}' @ \frac{2}{55}, & S \backslash NP: \text{eat}' @ \frac{2}{55}, & S \backslash NP: \text{no}' @ \frac{1}{55}, \\ S \backslash NP: \text{choc}' @ \frac{1}{55}, & S \backslash NP: \text{plu}' \text{veg}' @ \frac{2}{55}, & S \backslash NP: \text{plu}' \text{eat}' @ \frac{1}{55}, \\ S \backslash NP: \text{plu}' \text{no}' @ \frac{1}{55}, & S \backslash NP: \text{plu}' \text{choc}' @ \frac{1}{55}, & \\ S / NP: \text{veg}' @ \frac{2}{55}, & S / NP: \text{gone}' @ \frac{2}{55}, & S / NP: \text{eat}' @ \frac{1}{55}, \\ S / NP: \text{no}' @ \frac{1}{55}, & S / NP: \text{plu}' @ \frac{1}{55}, & S / NP: \text{plu}' \text{veg}' @ \frac{1}{55}, \\ NP: \text{veg}' @ \frac{9}{55}, & NP: \text{eat}' @ \frac{3}{55}, & NP: \text{plu}' \text{veg}' @ \frac{2}{55}, \\ NP: \text{plu}' \text{eat}' @ \frac{1}{55}, & NP: \text{plu}' \text{gone}' @ \frac{1}{55} & NP: \text{plu}' \text{choc}' @ \frac{1}{55}, \\ NP: \text{plu}' \text{no}' @ \frac{1}{55}, & NP: \text{no}' @ \frac{4}{55}, & NP: \text{choc}' @ \frac{3}{55}, \\ NP: \text{gone}' @ \frac{1}{55} & & \\ NP \backslash NP: \text{plu}' @ \frac{3}{55}, & & \\ NP / NP: \text{plu}' @ \frac{3}{55}, & NP / NP: \text{veg}' @ \frac{3}{55}, & NP / NP: \text{choc}' @ \frac{1}{55} \end{array} \right\}$$

Figure 8. The total set of hypotheses about the word *veggies* after three hypothetical scenes.

veggies := *NP:veg'* and *veggies* := *NP:plu'veg'*, rank highest, $\frac{11}{55}$, which are exponentially higher than almost all others. (Figure 8 is the source of these numbers.)

The plural is $\frac{4}{10}$ likely to mean *plu,'* which outranks all other alternatives except *veg,'* More experiences with the plural will give more diminishing returns for assumptions other than *plu,'*

More important to our present concern is *plu'*, which is $\frac{4}{19}$ likely to arise from *-s*, which outranks its competitors except *plu'veg'*, which is $\frac{5}{19}$ percent likely. The outranking hypothesis is associated with the word *veggies*. Together they embody a cross-situational parsed-to-learn understanding of the set {*veggies, -s*}, along with syntactic types. 75% of *-s*: *plu'* experiences are mediated by *NP \ NP*. The plural's possible connections to the hypotheses about *eat* in the first experience and *no* in the second one can only be indirect, that is, through some wrong assumptions about these words that they meant *veg,'* because otherwise they cannot be adjacent to *plu'*-assumed words. Its link to *gone* in the third experience is more direct because they are adjacent. This can be observed in *-s* types of (43). Its relation to the hypotheses about *veggies* is more involved, as can be seen from (41) and (43).

Once the *NP* hypothesis about *veggies* begins to win out, a Humean generalization of "something more than the experience" can be assumed to take place, where the winning strategy of typing *plu'* as *NP \ NP* and calling *veggie*-like things *NP* can come together in parsing other strings such

as *birds* and *doggies*. The types, in other words, conspire to relate certain bound meanings with certain free meanings once we have sufficient confidence in them. The other types for the plural and the noun would not be so successful *across experiences*. They are not winning strategies. This result comes from the interaction of Siskind's cross-situational inference and covering constraints, where the former sieves out the hypotheses by the intersection of scene meanings, and the latter eliminates some hypotheses by assuming that all hypotheses of an experience must be derived from the meanings of the words in an experience (we have somewhat relaxed this assumption but not much; in experience two there is no word for chocolate but some words were assumed to mean chocolate.)

Now we can be quite explicit about the form and substance of the linguistic knowledge of words: it is the set of categories it can bear, along with the owner's trust on the members of the set, acquired by the parse-to-learn paradigm. (Keep in mind that, for the purposes of this book, we ignore the aspects of morphology and inflection, such as *veggie* versus *veggies*, hence this is only a first approximation). The collection of such knowledge comprises an individual's grammar. For the hypothetical child above in particular, the collection might contain the fragment exemplified in Figure 8.

Notice that the knowledge of the child's word experience is complete. (This is a requirement for a computational model of the process, that the correct solution be on the search path even if it is not very likely at the beginning, since we know that every child converges on the competent use of a word after experience.) Her sums add up to 1, for both $\text{veggies} := \text{veg}'$ relation and for the possible categories of the word *veggies*. In this circumscribed and deliberately simplified world, the NP hypothesis for this word is the top contender after these three experiences, with a total frequency of $\frac{7}{14}$, in which the correct relation, $\text{veggies} := \text{NP}:\text{veg}'$ ranks highest, $\frac{3}{14}$.

One attempt to reduce the possible substantive categories in the search space of acquisition is the theory of functional categories, to which we now turn. The point of semantics in their case is that we do not need yet another innate source of knowledge for words, because although their semantics seem robust across languages, they are quite predictable as lexical items.

7. Functional categories

It is common practice in transformationalism to distinguish substantive categories such as V(erb), N(oun), A(djective) and P(reposition), from functional categories, such as C(omplementizer), I(nflection) and D(eterminer), among others. As the distinction has no place in radical lexicalism, one might wonder whether functional categories are quirky syntactic objects or arise from semantic dependencies.

The first thing to note about them is that they have a parasitic life. They depend on substantive categories. A determiner phrase (DP) needs a noun phrase, an inflectional phrase (IP) needs a tensed domain like root sentences, a complementizer phrase (CP) needs a clause, etc. We can narrow down our question to (i) whether these dependencies need combinators, and (ii) why they materialize in more or less the same way across languages when they manifest themselves.

Let us start with the last question first. Szabolcsi (1994) establishes the semantic bond across some apparently distinct functional syntactic items. Her *subordinators* are generalizations of nominal elements such as the article, the determiner and the verbal ones such as the complementizer. Their common function is to make the predicate or the nominal an argument of another predicate. For the nominal domain, say for the article, value-raising the article to take a noun and look for predicates looking for such arguments is a way to capture this behavior. Value-raised categories are those in which the result type (value) is a type-raised category, for example $(S/(S\backslash NP))/N$, rather than NP/N . On the semantic side it is accompanied by distributing type raising to the arguments, for example $\lambda P\lambda Q.(\forall x)imply'(Px)(Qx)$ for the quantifier *every*.

For the complementizer, it is usually the identity function, $\lambda P.P$. The difference seems natural without the need of a universal. Nominals are properties *and* arguments, whereas predicates as arguments do not engender another predication. There would be nothing over which value-raising could operate and distribute type-raising. We shall see that once we translate functional distributional categories to combinatory ones, they have nonvacuous but semantically transparent functions such as $\lambda P.P$.

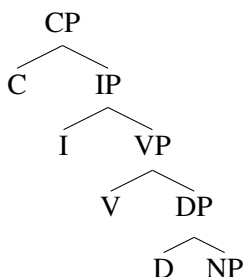
Regarding the first question, whether we need combinators for functional categories, we can start with the original motivation for positing functional categories: the substantive-functional distinction is meant to capture lexical-universal structures. Functional projections, as the theory goes, always bind

the substantive phrase in the same way, whereas the relations within a substantive phrase can be language-particular.

Grimshaw (2000) is a summary of the developments and the universal claims about functional categories (see also Pollock 1989, Haegeman 1998 for more functional categories). Her formulation is a good starting point to see the possible dependencies, and we can assume a version of it to be part of a meta-theory for predicting possible lexical category-feature mappings in CCG. (CCG would be overextending itself to cover cases where the configurations are not syntacticized by combinatory dependencies. In this sense, it needs meta-theories such as this and for example autosegmental phonology. But we must first be sure that the dependencies are not universal but lexical.)

Among the possible projections Grimshaw reports, the one in (44) is perhaps the most expected, which summarizes the motivation for the idea of distinguishing functional heads (C, D, I) from lexical heads (N, V, A, P). In the text the C-IP head-complement relation is bracketed as [C IP]_{CP}.

(44)



Other possible head-complement configurations according to her are C-VP, P-DP and P-NP. The impossibility or oddity of some of the configurations according to Grimshaw, such as I-DP, V-IP, D-DP, C-VP, I-NP arise from her theory of projecting lexical heads only under the guidance of functional heads.

VP is a lexical projection in (45a) whereas IP is a functional projection dominated by it, which is considered illicit. This is impossible according to Grimshaw because of the functional mismatch in VP and IP, although there is a categorial match between V and IP, say as [+V -N]. (45b)'s violation is considered less severe because there is no categorial match in V and DP, hence an ambiguous extended projection is expected.

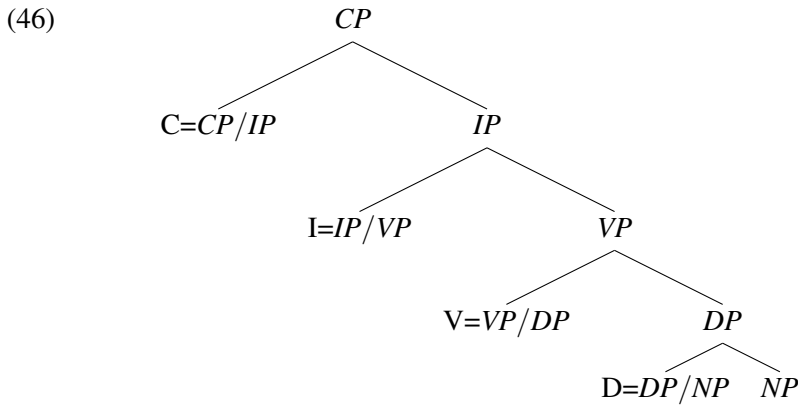
(45) a.



b. ?VP



From the perspective of heads, their combinatory categories can be given the following first approximation in association with (44).



Take the category of C, viz. CP/IP . To say that IP is an inflectional projection (e.g. S_{fin}) is to categorize the complementizer as S'/S_{fin} , as we have so far assumed for example for *that*, as in *I think that she likes me*, rather than S'/S . A category such as S/S_{fin} does not capture CP/IP either. IP cannot be an agreement domain *typewise* because, in the domain of locality of C, that is, in its lexical category CP/IP , such as S'/S_{fin} , there is no argument to agree with. (Structure-wise it can have an agreement element in it such as INFL, in theories that posit functional categories. Agreement as a type domain cannot rely on this property. Types are string properties, not tree properties.) Semantically the complementizer translates to $\lambda P.P$ since there are no arguments or predicates whose dependencies must be heeded.

Consider now the category of I in (46), IP/VP . Positing this category is the same as saying that all arguments are type-raised in a competence grammar, either lexically or by a lexical rule, so that categories *onto* IP must heed agreement, for subject-agreement languages. (Note that this is not a universal, e.g. Chinese).

We can then follow the influential proposal of George and Kornfilt (1981) to take finiteness as a corollary of agreement, for both verbs and nouns; see Kornfilt (1984), Abney (1987). For English it means *she* in *she likes chocolate* bears the category $S/(S \setminus NP_{3s})$, not just $S/(S \setminus NP)$, and *likes* bears $(S_{\text{fin}} \setminus NP_{3s})/NP$, not just $(S \setminus NP_{3s})/NP$ or $(S \setminus NP)/NP$.

It also means that *her* in *she likes her* must not bear such decorations although it carries morphologically the number and person, e.g. $S \setminus (S/NP)$. Notice that $S \setminus NP$ is VP whereas S/NP is not, thus what we have captured

lexically is the essence of *IP/VP*. The agreement domain in English is $S \setminus NP$. (For Welsh, which is strictly VSO, the difference in agreement domains and others can be accounted for by $S \setminus (S/NP_{3s})$ for subject and $S \setminus (S/NP)$ for non-subject third-person NPs.) The semantics of the process involves no freely operating combinator; it is the semantics of lexical **T**, for example $Joe := NP_{3s} : joe' \rightarrow S / (S \setminus NP_{3s}) : \lambda P.Pjoe'$

Now consider the substantive category *V* in (46). It gets a functional interpretation in structure-dependent theories because of its licit configuration $[V-DP]_{VP}$, which we could translate as $V=VP/DP$. In CCG, it amounts to saying that the *DP* is a nonagreeing argument because *VP* is the domain of agreement, not *DP*, which we can capture as $(S \setminus NP_{agr})/NP$ in *V*'s category for English. (For Welsh, the category is $(S/NP)/NP_{agr}$ because the first NP is the subject.)

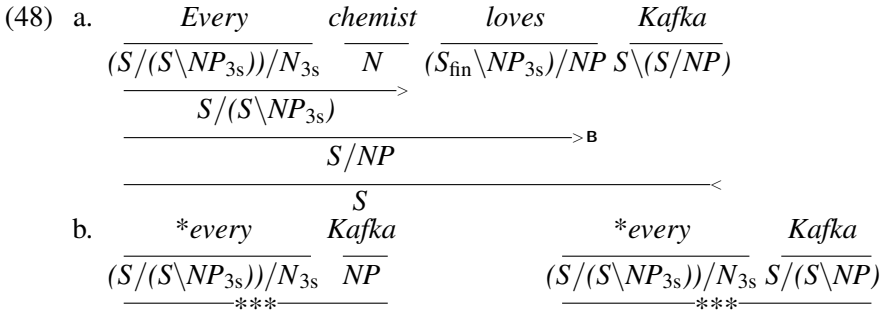
The mutual dependence of *VP* and *IP* on *V* in distributional-category theories is captured in combinatory categories by the fact that all the arguments are type-raised, and they can differ in agreement. Thus the *V-DP* configuration turns out to be a lexical category, viz. $(S \setminus NP_{agr})/NP$ for English. As *V* is a substantive category in everybody's theory, it follows that its category is not universal, for example $(S \setminus NP)/NP : \lambda x \lambda y.read'xy$ for the SVO English and $(S/NP)/NP : \lambda x \lambda y.read'yx$ for the VSO Welsh.

Finally, let us consider the functional category *D* in (46), which translates to DP/NP . This conception of *NP* must be headed by an *N* rather than a determiner. Thus we have DP/N in categorial terms. Considered together with the *DP* category mentioned earlier, the DP/NP assumption amounts to saying that all determiners, including quantifiers and names, are type-raised or value-raised, since *DP* necessarily functions as an argument (the *N-DP* configuration is illicit in functional projection theories as well).

The idea has been around since Russell and Montague (1973) as the theory of generalized quantifiers. For example, the categories in (47a) handle (47b), where determiner- and name value-raising (and concomitant differences in agreement) also handle (47c–e) (assuming *Kafka* is a name, not a property). These are shown in (48).

- (47) a. $every := (S / (S \setminus NP_{3s})) / N_{3s} : \lambda P \lambda Q. (\forall x) imply'(Px)(Qx)$
 $every := (S \setminus (S/NP)) / N_{3s} : \lambda P \lambda Q. (\forall x) imply'(Px)(Qx)$
 $Kafka := S / (S \setminus NP_{3s}) : \lambda P.Pkafka'$
 $Kafka := S \setminus (S/NP) : \lambda P.Pkafka'$
 b. *Every chemist loves Kafka.*

- c. *Every chemists love/loves Kafka.
- d. Kafka loves/*love every chemist.
- e. *every Kafka



As expected, the semantics of D cannot be due to a syntactically operating combinator. (Note that *x* and *kafka'* in (47) are not syntactic variables.) The differences all lie within the lexical syntactic type restrictions.

Let us now consider some of the impossible configurations which the functional-category theory rules out by purely formal means. Take Grimshaw's I-NP and D-DP. Assume an as yet undetermined projection for I-NP, say *XP*. We could categorize I as *XP/NP*. To be faithful to the semantics of inflection, which 'I' stands for, we must obtain an agreement range. No type for *XP* can deliver this interpretation. Take *XP=S*. Then *S* would not be an agreement range. Take *S_{agr}* for *XP*. Then the *XP* of *XP/NP* must be *IP*, but the *IP* domain requires type raising of all arguments, and *IP/NP*, which would be *S_{agr}/NP* in the current assumption, would not be type raising.

Now consider D-DP, where *D=XP/DP*. Since *D=DP/NP* is possible, we get *XP=DP* and *DP=NP*. The last one is the standard assumption in CCG. But *XP=DP* would predict overquantification because *D=XP/DP=DP/DP*. The structural equivalent of this assumption would be $[D[D\ NP]_{DP}]_{DP}$. This assumption, *XP=DP*, cannot capture the semantics of quantification because there would be no discernible head for *DP*.

In summary, what is called a functional category is in essence (i) a syntactic restriction on grammatical meanings which narrows down the compositional meanings that must be delivered by a competence grammar, and (ii) a faithful reflection of semantic headness on syntactic types. Functional categories need not be ordained as special combinatory rules, or special categories, because they do not engender semantic dependencies that must be captured by a syntactic combinator. Thus there is nothing special about them

that a lexical category cannot handle; they all belong to the lexicon. They are special in the sense that they form a closed set, for example, every language seems to have a universal quantifier, a finite set of determiners (maybe none), a small set of complementizers, a fixed inventory of case markers etc. But adpositions and pronouns form a closed class as well, hence this is not their definitive feature.

The choice of a basic category inventory including the functional ones interacts with accounts of constituency. For example, if complement clauses are S rather than S' , we would be hard pressed to eliminate (49a) while accounting for (49b).⁹⁰

- (49) a. $*[I \text{ think that Harry}]_{S/(S \setminus NP)}$ and $[Barry]_{S/(S \setminus NP)}$ like Mary.
 b. $[I \text{ think that Harry}]$ and $[Barry \text{ thinks that Mary}]$ owns the house.

A combinatory theory would be overextending itself if it chooses to eliminate (49a) by some combinatory restriction. The problem does not arise from the category of *that*, which is already onto S' , typically assumed to be S'/S or S'/S' . It is the category of *Harry likes Mary* as a complement clause, which, as S , leads to the problem above. If we can type-raise the embedded subject *Harry* as $S'/(S' \setminus NP)$, the problem disappears because the conjuncts in (49a) would not be like-typed for that interpretation:

$$(50) \quad \frac{\frac{I}{S/(S \setminus NP)} \quad \frac{\text{think} \quad \text{that} \quad \text{Harry}}{(S \setminus NP)/S' \quad S'/S' \quad S'/(S' \setminus NP)}}{S/(S' \setminus NP)}$$

Then we have to find an empirical justification for typing the subordinate verbs to be onto S' rather than S , e.g. $(S' \setminus NP)/NP$ for *like* and *owns* above. The syntactic aspect of the justification is clear: these are not main clauses.

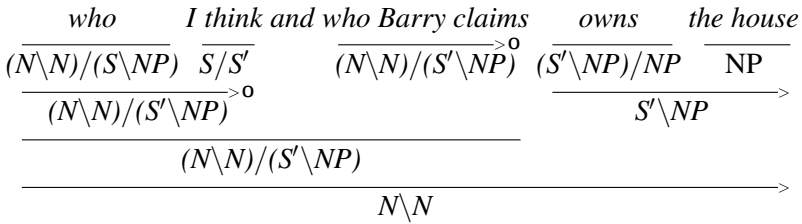
This may be a good move in English syntax to be able to account for examples such as the following without further assumption:

$$(51) \quad \frac{\frac{\text{the man who } I \text{ think and Barry claims}}{S'/S'} \quad \frac{\text{owns}}{S'/S'} \quad \frac{\text{the house}}{(S' \setminus NP)/NP} \quad \text{NP}}{[S'/S'] \quad [S'/S'] \quad \xrightarrow{\hspace{10em}} S' \setminus NP}$$

I write the standard assumptions in square brackets (i.e. if we assume S as a result, not S') and the new ones on top to show that it is not the result but the domain type of substrings such as *I think* that we should worry about because either assumption would give us a residue as a function from S' to something.

Further support for a subordinate category such as $(S' \setminus NP) / NP$ for the subordinate verb, and also for the presence of **O** in English syntax, comes from the following example which works with the standard assumptions for everything else:

(52) *the man*



German and Turkish show that the degree of freedom here is still within a radically lexicalized grammar: distinct word order in subordinate clauses of German, in contrast with second-position verbs in main clauses, and distinct Turkish subordination morphology, where word order for subordinate clauses is the same as main clauses but morphology differs; the subordinate subject and the verb must carry overt agreement morphology which is distinct from main clause agreement morphology. In other words, an external constraint or rule is not necessary.

The functional categories seem to have in common the semantic property that they operate over PADSs in which the predicate is always opaque, as in the type-raising of arguments, value-raising of properties and participants, and complementizer semantics. They cannot latch on to a substantive meaning directly. Radical lexicalization makes this aspect very explicit due to forced syntax-semantics correspondences in a lexicalized grammar.

The theory of functional categories can be seen as a quest for more refined restrictions on lexicalized syntactic types, and also as an aid in search of good bootstrappers for learning. Brent (1993) shows how far the idea can go in computational learning of lexicalized grammars in an unsupervised way, with a warning that it needs a narrowly constrained theory of possible grammars. The closed set of items does the work of self-supervision. There seems to be the correlates of these assumptions in the acquisition environment of the child. We know that children are late in producing function words, but they seem to zoom in on them early in analyzing utterances (Santelmann and Jusczyk 1998), and the frequency of function words is consistently higher than the frequency of content words, both in child-directed speech and in adult speech, across languages (Shi, Marquis and Gauthier 2006).

8. Case, agreement and expletives

Some other special categories that serve functionally without apparent semantic content shows characteristics similar to that of functional categories.

Any lexical functor that has an argument (say an NP) in its domain of locality can refer to its consistently discernible features, such as case, agreement, noun class, tone (for tone languages) and locus (for signed nouns). There would be no basis for the functor to look at a nondiscernible feature, such as whether it modifies a noun that starts with the phoneme /b/, since that information cannot be coded in syntactic types.

A list of typical functors can give us an idea about agreement controllers:

- (53) a. verbs, e.g. $S \setminus NP, (S \setminus NP) / NP$
 b. adjectives, e.g. N / N
 c. nouns, e.g. $N / (N \setminus N), N$
 d. determiners, e.g. $(S / (S \setminus NP)) / N$
 e. relative pronouns, e.g. $(N \setminus N) / (S / NP)$
 f. prepositions, e.g. $(N \setminus N) / NP$
 g. adverbials, e.g. $(S \setminus NP) \setminus (S \setminus NP)$

Examples of agreement involving these functions include: subject and verb (Portuguese), subject, object and verb (Uralic languages), adjective and noun (Russian), noun and noun in possessor constructions (Georgian), determiner and noun (German), relative pronoun and noun (Latin), preposition and object (Welsh), adverbial and subject (North Caucasian languages). Thus all possibilities that are allowed by functor types are attested for argument-taking entities, and they cross-cut the accusative-ergative-split classification of languages and word orders.

The radical lexicalization of functional categories (§7) suggests that all these patterns are lexicalizable, and the lexical combinatory categories that arise out of these considerations clearly distinguish agreeing and nonagreeing arguments. Take for example some quirky cases of agreement. The combinatory nature of the domain of locality and type raising of arguments facilitate a natural account of what is called “brother-in-law agreement” in Relational Grammar (Perlmutter 1983), exemplified below.

- (54) a. *There are/*is cows in the garden.* Aissen (1990)
 b. *There seem to be some bugs in the soup.* Perlmutter (1983: ex.65)

Since this is not triggered by the copula but by the expletive, it follows that the category of the expletive must take the raising verb as an argument *first*, and value-raise it, which provides a domain of locality where all the agreement features, including that of the NP following the copula are available to the expletive.

We can think of raising verbs as forming a typewise discernible class in the lexicon. Following Clark (1997), Steedman (2000b), I will consider the auxiliaries and the copula as raising verbs (55).

- (55) The class of raising verbs: (V_{rz})
 $are := (S \setminus NP) / (S \setminus NP_{agr}) : \lambda P \lambda x.be'(Px)$
 $might := (S \setminus NP) / (S \setminus NP) : \lambda P \lambda x.might'(Px)$
 $seem := (S \setminus NP) / (S_{to-inf} \setminus NP_{agr}) : \lambda P \lambda x.seem'(Px)$

Raising verbs such as *seem* follow the same pattern in their dependency structure. Note however the lexical differences, such as the brother-in-law agreement for the copula and *seem*. I will collectively refer to them as V_{rz} . Their common pattern in the PADS is the crucial aspect of the generalization.

A single lexical category for the expletive *there* is sufficient to handle brother-in-law agreement, without the necessity to posit another agreement pattern. As this is lexically triggered by the expletive, it would have no relation to the object-agreement systems of ergative languages.

All NPs in the locality of the expletive have the same agreement information in (56a), which yields the right behavior in (56b–c).

- (56) a.

<i>There</i>	<i>are</i>	<i>cows</i>	<i>in the garden</i>	
$S /_{\star} ((S \setminus NP_{agr}) \setminus ((S \setminus NP_{agr}) / NP_{agr}))$	$V_{rz,plu}$	$(S \setminus NP_{agr1})$	$(S \setminus NP)$	$<$
$: \lambda P \lambda Q.Q(P self')$	$: \lambda P \lambda x.be'(Px)$	$\setminus ((S \setminus NP_{agr1}) / NP_{plu})$	$\setminus (S \setminus NP)$	
$: \lambda P \lambda Q.Q(P self')$		$: \lambda f.f cows'$		
$S /_{\star} ((S \setminus NP_{plu}) \setminus ((S \setminus NP_{plu}) / NP_{plu}))$				
$: \lambda Q.Q(\lambda x_1.be' self' x_1)$				
$(S \setminus NP_{agr1}) \setminus ((S \setminus NP_{agr1}) / NP_{plu})$				$<B$
S				$>$
$: in' garden' (be' self' cows')$				

b. *There is/*are a cow in the garden.*
c. **There is/are.*

In other words, *there* not only equalizes argumenthood in semantics (see its PADS, where the predicate P is reduced on *self'*), it also equalizes agreement in syntax by underspecification (see its agreement features, which are all agr).

This stands in contrast with the type raising of all other subjects in English, which all carry an agreement constraint, for example $S/(S \setminus NP_{3s})$ for *she*.

Radically lexicalizing the neutralization of agreement also gives us an opportunity to account for the following difference, where *there's* is another lexical item:⁹¹

- (57) a. *There's many people here.*
 b. **There is many people here.*

The expletives are quite idiosyncratic (*it* is not a neutralizer; cf. 58a–b). Thus lexical value-raising of the brother NP by the expletive is justifiable (value-raising is needed to get the right PADS, and the corresponding propositional type for *Q* is required by the Principle of Categorical Type Transparency).

- (58) a. *It is/*are important that we call the cows home.*
 b. *It seems/*seem to rain. There seems/seem to be a problem.*
 c. **There is himself/herself in the garden.*

The predicate-argument structure of *there* ensures that the brother NP becomes the maximally PADS-commanding argument (without a linking or chain theory), which is consistent with the ungrammaticality of (58c), assuming of course a genuine reflexive reading. Because of the universal nature of binding, we would expect all languages with brother-in-law agreement expletives to follow (58c).

The argument depends on the assumption that all arguments are type-raised in competence grammars. We can see the empirical consequences of this in the following example, where a participant (i.e. type-raised) category is acceptable but a property is not.

- (59) *Who's going to help me do the dishes?*
 Well, *there* *is* *John* */*man.*

$$\frac{S/\star((S \setminus NP_{agr}) \setminus ((S \setminus NP_{agr})/NP_{agr})) \quad (S_{be} \setminus NP_{sg})/NP}{\begin{array}{l} / \star((S_{be} \setminus NP_{agr})/NP_{agr}) \\ : \lambda P \lambda Q. Q(P \text{ self}^t) \end{array}} : \lambda x_2 \lambda x_1. be' x_2 x_1} \longrightarrow \frac{S/\star((S \setminus NP_{sg}) \setminus ((S \setminus NP_{sg})/NP_{sg}))}{: \lambda Q. Q(\lambda x_1. be' \text{ self}^t x_1)}$$

Notice that the category of the expletive does involve type raising, just like other subjects, and the copula agrees with the subject, just like other verbs. We are not setting up a separate expletive syntax, or a special nonthematic

role for the expletive for the verb to worry about. The expletive's uniqueness is to take a type-raised brother NP category as an argument so that it will have lexical access to the domain of locality of that NP. Without this, we could not claim to have captured the competent knowledge of the expletive, because the examples below could not be handled. Thus the competent knowledge of the expletive presumes the knowledge of type raising in the language.

- (60) a. *There are cows in the garden and mice in the kitchen.*
 b. **There are cows in the garden and a mouse in the kitchen.*

The expletive is the only exception to type raising of subjects, in languages with expletives. We can conjecture that expletives are acquired quite late, after many syntactic environments have been encountered, giving enough exposure for type raising of objects to be mastered.

The point of the expletive's category is that, if we are to account for its unique agreement behavior and argument-taking, we cannot simply rely on the presence or absence of thematic roles; we must show a PADS that arises from syntax like everything else. Its semantics cannot be empty (witness the PADS in (56a) which includes a substantive component *self'*), unless we set up a special syntax for the expletive. That of course is not the agenda of radical lexicalization.

9. The semantics of scrambling

The radical lexicalization of functional categories (§7, §8) as part of a theory of feature geometry suggests a clear distinction between agreement and nonagreement domains of type-raised arguments. We can expect subject-agreement languages to type-raise the subject in ways that enforce agreement. Likewise, we can type-raise an accusative NP to an agreement domain if there is object agreement in the language, as in Uralic languages.

Since type raising is order-preserving, and its liberal variety in syntax would be devastating because of permutation closure (Moortgat 1988a), we will not get free word order just because all arguments are type-raised. These results suggest that free word order must be a conspiracy of more than one grammatical resource. Steele (1978) clearly shows that it cannot be just case marking because some languages with morphological case show no sign of scrambling (e.g. Albanian), and some languages without case allow it (e.g. Classical Aztec, Garadjari).⁹² A freely permuting verbal category or some

stylistic (exogrammatical) choice cannot be the answer either because so-called scrambling languages do impose limits on it, and when it is licensed, every different order seems to add some information-structural aspect to the PADS.⁹³

The key point that forces us to keep so-called scrambling in grammar—therefore do something about its semantics—is that, although there can be multiple factors to induce a permuted sentence, all the resources involved relate to grammar: case, morphology, intonation, information structure, an attempt at the disambiguation of scope, etc.

Take for example the following sentence pairs from a so-called scrambling language. Example (61a) is ambiguous; there can be more than one car. (61b) however is not ambiguous.

- (61) a. *Her çocuk araba-ya bin-di.* Turkish
 every child car-DAT mount-PAST
 ‘All children went in the car.’
 b. *Arabaya her çocuk bindi.*

The unambiguity of (61b) is not forced by word order alone. There are presuppositions, for example, that all the children were waiting. If some children have taken the train, ending the event of train-taking, we are back to an ambiguous interpretation. A competence grammar should deliver both readings, which are different semantically to begin with, and an oracle must choose between them depending on context and intonation.⁹⁴ The oracle is going to need some grammatical information to disambiguate, and the delivery of that information is the grammar’s responsibility. Thus radically lexicalized grammars must deliver different things about different word orders, otherwise the grammar itself must be the oracle. This would fly in the face of radical lexicalism because it amounts to saying that all contextualizations must be lexicalized in the grammar, a result which seems theoretically possible but very unlikely.

A competence grammar of Turkish must also handle the apparent asymmetry caused when the same process of word order flexibility is repeated postverbally. In the examples below, we are not forced to think of elaborate alternatives or presuppositions to see that both are ambiguous. Kornfilt (2005), Kural (1994, 1997) concur with these observations.

- (62) a. *Bindi her çocuk arabaya.* Turkish
 b. *Bindi arabaya her çocuk.*

The postverbal process seems language-specific, suggesting a lexicalized solution to the syntax-phonology interface, rather than some universal. For example, a Russian speaker could say *Denis udaril Sashu* to mean either ‘Denis hit Sasha’ or ‘It was Sasha that Denis hit’, but a Turkish speaker would never use this word order to convey the second reading.

This facilitates a minimal comparison of alternative grammars to see for example the interaction of the semantics of the accusative case and the category of the verb. The Turkish verb must be typed head-final in the lexicon to account for the contrast in (61) and (62), otherwise case marking itself cannot deliver information about head-finalness of surface word order to an oracle. The reason is as follows. If we categorize the transitive verbs as $S\{|NP_{nom}, |NP_{acc}\}$ to handle all variations on word order, where the set notation indicates arguments in any order (following Baldrige 2002), a backward type-raised accusative cannot be assumed to take the role of indicating a postverbal order; both orders below would be fine with that type:⁹⁵

- (63) a. $\frac{\frac{\textit{her } \textit{\u00e7ocuk} \quad \textit{\u00e7ukulata-y\u0131} \quad \textit{sev-di}}{\text{every child.NOM} \quad \text{chocolate-ACC} \quad \text{like-PAST}}}{\frac{S/(S/NP_{acc})/(S/NP_{acc} \setminus NP_{nom}) \quad S \setminus (S/NP_{acc}) \quad S\{|NP_{nom}, |NP_{acc}\}}{S/(S/NP_{acc} \setminus NP_{nom})} <_{B \times}}$
- ‘All children liked (the) chocolate.’
- b. $\frac{\textit{sev-di} \quad \textit{\u00e7ukulata-y\u0131} \quad \textit{her } \textit{\u00e7ocuk}}{S\{|NP_{nom}, |NP_{acc}\} \quad S \setminus (S/NP_{acc}) \quad S \setminus (S/NP_{nom})} <_{B \times}$
- S/NP_{nom}

The verbal category must be revised to fix this. If we assume Turkish is head final, i.e. the transitive verb is of type $S\{\setminus NP_{nom}, \setminus NP_{acc}\}$, then backward type raising cannot derive (63). Forward type raising cannot help with the asymmetry either because it cannot deliver (63b) in the first place. Now we must call in another resource, which we must relate to intonation because we have used up other resources. In a radically lexicalized grammar, this must arise from a lexical category, which has been identified as the lexical rule for rightward contraposition by \u00d6zge and Bozsahin (2010):

(64) $NP \rightarrow S_{\beta} \setminus (S_{\beta} \setminus NP_{\beta}) \quad (\beta \text{ for background}) \quad (>_{T \times})$

The rule says that all nominals, irrespective of their case, yield a different kind of sentence when they are backgrounded, to deliver a rheme- or theme-backgrounded clause. The proposal is purely type-dependent, not position or structure-dependent, because it simply correlates an exclusively backward-

looking category with backgrounding. The β feature is reflected on the PADS objects as a side effect, by marking them background rather than more salient or contrastive. Because of the result's directionality in (64), it can only combine arguments that are postverbal, which indirectly (i.e. grammatically) associates postverbalness with backgrounding in Turkish. Thus we have all the information to be delivered at the interfaces to communicate the informational differences between (63a) and (63b) via their PADS:

- (65) a. $\frac{\textit{her } \textit{\c{ocuk}} \quad \textit{\c{ukulata-yi}} \quad \textit{sev-di}}{S/(S \setminus NP_{\text{nom}}) (S \setminus NP_{\text{nom}})/(S \setminus NP_{\text{nom}} \setminus NP_{\text{acc}}) S\{ \setminus NP_{\text{nom}}, \setminus NP_{\text{acc}} \}}$
 $\xrightarrow{\text{B}}$
 $\frac{S/(S \setminus NP_{\text{nom}} \setminus NP_{\text{acc}})}{S\{ \setminus NP_{\text{nom}}, \setminus NP_{\text{acc}} \}}$
 b. $\frac{\textit{sev-di} \quad \textit{\c{ukulata-yi}} \quad \textit{her } \textit{\c{ocuk}}}{S\{ \setminus NP_{\text{nom}}, \setminus NP_{\text{acc}} \} S_{\beta} \setminus (S_{\beta} \setminus NP_{\beta, \text{acc}}) S_{\beta} \setminus (S_{\beta} \setminus NP_{\beta, \text{nom}})}$
 $\xleftarrow{\text{B}}$
 $S_{\beta} \setminus NP_{\text{nom}}$

Now we can clarify the semantics of the accusative case which in some accounts is assumed to be vacuous. It cannot be directly information-structural or about definiteness, because such matters are not always lexicalizable. Witness (63a–b), where the accusative NP is not necessarily definite.⁹⁶ It is not necessarily a theme or rheme either. Therefore a lexical category for the accusative marker must be neutral, i.e. it must be $\lambda P.P$, which by definition makes P predicational.

What makes P a dependency arising from a transitive verb is its syntactic type, not its predicate-argument structure. It can be indirectly information structural, as in (64), which presupposes that it has a PADS to begin with so that an update on that PADS can take place. Since the syntactic type of the accusative forcibly faces a $\lambda P.P$ semantics, it has no room for substantive side-effects. It can only pass down the informational features, which must be put in the PADS and the syntactic type by other items. That is why the accusative can only be a projector of informational features, rather than being an instigator. See Özge (2010) for more arguments supporting this conclusion.

Everything in the lexicon must have a PADS, i.e. semantics, otherwise we cannot account for interactions between the lexical categories. That is, we cannot create a grammar.

10. Searle and semantics

Assuming that there must be some kind of semantics in the grammar, and that the kind of semantics we can put in the grammar must be compositional for syntax to do its work, we can question whether this semantics is just another name for syntax, or for formal symbols. The issue relates to the longstanding argument that syntactic manipulation alone cannot give rise to meaning.

Searle (1980) in his Chinese Room thought experiment sets out to show that a purely formalist account of the mind is not possible. It relates to our present concern because he chose language, in particular semantics, to make his case. The specific claim he was arguing against is strong AI, the claim that a functional interpretation of the mind counts as a mind. This view according to Searle is bound to fail in its aspirations because the kind of computation it envisages is formal, i.e. it operates over symbols with no content, whereas the mind sets up, he claims, relations between intentional states and the world, via causal powers of the brain. We must have “the right stuff,” i.e. a human brain, to have that causal power, according to Searle.

In the same article (and subsequently in 1990a), Searle addresses possible objections to his claim, which are mainly concerned with what is embodied in the Chinese room. Searle called them “the system’s reply”, “the robot reply”, “the brain simulator reply”, “other minds” and “other mansions” reply, and their combination, against which I believe he defends his position quite convincingly.

Recall also some other criticisms, such as Rey’s (1986) argument that mental states are species-specific for all species anyway—which to me suggests that ascribing semantics to certain states of a machine ought to be constructed by the machines, and the experience cannot be presided over by an external judge.

Rey’s argument I think brings a Husserlian perspective into the debate in which we can talk about sharing the subjective experiences of humans *among themselves* but most likely not with cats or ants, which leaves open the possibility that they can do the same thing and do not inform the humans about it. It amounts to saying that the mental states can be real for all species. With a stretch of imagination we can grant the same ability to machines that perceive, act and react. However, I will not follow this line of argument.

It is interesting that the debate continued between Searle, the philosophers, psychologists and AI researchers, with almost no argument from linguistics (but cf. Carleton 1984). I offer one in this section from philosophy of linguis-

tics, to question whether the Chinese room as imagined by Searle is possible. My argument is about what Searle considers computational, and about the linguistic conception of the same notion, which must, according to many cognitive scientists, indirectly relate to semantics.

First a summary of Searle's argument, from a more recent self statement (Searle 2001): imagine a native speaker of English, who has no knowledge of Chinese, locked in a room full of boxes of Chinese symbols (a database) together with a book of instructions written in English (the program), which he can interpret, for manipulating the symbols. More Chinese symbols are sent in to the room (questions), which the person in the room correctly answers in Chinese symbols by following the instructions for matching the database symbols and the symbols in questions. The person passes the Turing (1950) test in communicating Chinese, that is, a native speaker of Chinese at the other end of the box cannot tell that the answers are not coming from a Chinese speaker. Yet the person in the room does not understand a word of Chinese. The program and the database add no understanding of Chinese to the person, though he already knows how to interpret symbols in one language, namely English. By extension, computers cannot understand Chinese (or any human language) by purely formal manipulation of symbols.

The linguistic aspect of the experiment I think is as follows: what is Chinese in the Chinese room is the database and the fragments of the program that contains Chinese symbols and their abstractions (the program is in English, but it is about Chinese symbol correspondences). The program cannot be of infinite size (otherwise it would not be a program), therefore the correspondences in the program cannot be phrase-to-phrase matchings, for we can conjecture that there are potentially countably infinitely many Chinese expressions. (Or, if we take the infinitude claim to be less critical for language, as I have argued to be the case in §3.3, then we can say that the competent speaker's knowledge of phrase-to-phrase matchings would be too large to fit into any room.)

Hence the program must contain finitely characterizable symbols and their program-internal abstractions, such as calling a group of symbols a certain kind of category, and certain combinations of categories to be other categories, and so on and so forth, in other words, a grammar of Chinese. It does not matter for our current purpose that such a grammar is not necessarily lexicalized; its finite representability is the key point.

In the thought experiment we *must* assume that the program contains a (competence) grammar because we can "suppose also that the programmers

get so good at writing the programs that from the external point of view—that is, from the point of view of somebody outside the room in which I am locked—my answers to the questions are indistinguishable from those of *native* Chinese speakers.” (Searle 1980; my emphasis).

Let us now turn to the boxes of Chinese symbols. They would minimally contain Chinese vocabulary, and perhaps more, such as a large inventory of expressions based on symbols in the program. This too must be finite to fit into the room. We thus have a system of grammar and a lexicon housed in the room.

I claim that the experimental setup is inconsistent because of the forced assumption of housing a grammar, and not being able to use it for semantic interpretation. All grammars in any linguistic theory are interpretable because their product is there solely to provide a full array of phonetic, semantic and syntactic interpretation. The theories only differ on how they go about getting these interpretations from a surface string, and how to explain them.

What, then, is the problem with computation in Searle’s program? In the linguistic sense, the program is not doing computation at all, because computation is what links the string (the phonological form) and the meaning (say the PADS) at the interfaces to perceptual and conceptual systems of cognition. The link is the critical assumption, and needs further refinement.

In the Minimalist Program of Chomsky (1995), computation is conceived as the operation that links the stages of deriving a surface string, where intermediate results as syntactic objects are kept for later use. It seems to me that Searle’s choice of natural language computation for his thought experiment is inspired by an interpretation of Chomsky in an early incarnation.

Chomsky nowadays maintains that the interpretation of the string begins after its features are delivered to spell-out, at which point its access to lexicon—hence to meaning—is cut off, and the string is ready to be pronounced. More specifically, Searle seems to have in mind what Brody (1995) later called radical minimalism, where the phonological form is just an interpretation of a *single* interface, and the semantic interpretation rules and the lexicon have access only to that interface. This seems to be a more literal implementation of having a single hole in the box for outside access. This might appear to suggest that what takes place is essentially formal symbol manipulation of the morphological or phonological kind.

This is not entirely correct. Interpretable features are always carried *within* the intermediate records of syntactic objects. This was true in the pre-spell-out period of Chomsky as well, under different guises. I am in no position to

defend the Chomskyan view of carrying the semantics along, but clearly we do need room for these features in a faithful thought experiment of syntactic rule manipulation.

Moreover, we have seen that syntax and semantics can be derived in lock-step so that they are available at any time. For this to work in the Chinese room, semantics must be allowed to enter the room as well rather than expected to rise in it. Radical lexicalization shows that these meanings will arise only from the meanings of the words in the string because there are no intermediaries, and the semantics of common dependencies is invariant.

Therefore the Chinese room as a whole must have access to strings *and* meanings outside the room to be able to hypothesize about internalized meanings. Marconi (1997: 137) raises a similar objection: “a meaningless linguistic symbol cannot be made meaningful by being connected, in any way whatsoever, to other uninterpreted symbols.” It appears then that Searle is arguing from one conception of language computation, which is not universally shared (and might be considered dubious by its practitioners), to show that syntax suffices to legitimize his picture of the Chinese room.

What takes place in the room is not computation in the computing science sense either, for that computing is a link too, to link the programs (the form) with the executable code (the meaning), at the interfaces of the machine to the programmer’s expressions and intended tasks, the latter of which cannot be determined by the computational system.

We are reminded of Searle’s (1990b) claim that running the *wordstar* program might as well be undertaken by the wall behind him, since the wall is complex enough to embody the formal structure of *wordstar*. This is a gross oversimplification of computation. Programs execute only when they are interpreted by the “right stuff”, which is *in their case* a virtual machine instruction set. If the wall has the right stuff, then surely it can execute *wordstar*, but then it would be a brick-implemented computer rather than just a wall. Rey’s (1986) warning that strong AI is not behaviorist but functionalist makes the same point. I am not defending strong AI against Searle, but it need be said that he faces the same oversimplification of rule-following in computation as he does in syntax.

An uninterpretable program has no semantics—it is not a program, whereas a program that *does* nothing has one, with perhaps free interpretation in the programmer’s world. Thus Searle’s criticism of formal symbol manipulation as the basis of understanding may be directed towards possible reductionism of some programmers doing nothing but syntax, or for not

showing anything interesting in the way of semantics in current practice, but it is not an intrinsic problem of computation.

One might argue that semantics as conceived above is not really semantics because it is not situated in the external world, but this is precisely the point in linguistics and computing: language-internal semantics is only a gateway to the conceptual system, then to the world, where meaning cannot be determined by language. Language provides a semantic representation over which external (anchored) meanings can be enumerated. That is, understanding is an interface problem of connecting internal and external meanings for all kinds of species, natural or artificial.

Melnyk's (1996) objection to Searle follows a similar line of thought for programs. Marconi's (1997) point about inferential and referential knowledge of words as lexical competence, independent of whether the doer of symbol manipulation is natural or artificial, carries the same message: "The genuine problem is not whether knowledge of meaning can be "reduced" to symbol manipulation but what kind of symbol-manipulating abilities would count as knowledge of meaning or understanding of language" Marconi (1997: 137).

If this is the case, then a computational system can in principle be made to face the same conditions as the child for understanding the connections between sounds and meanings, once we readjust our semantic radar and incorporate compositional meanings into the notion of category.

There is already some progress in the way of breaking the "semantic divide" of a child's acquisition of language and a computational learning of human language. Zettlemoyer and Collins (2005) experiment with statistical learning of grammars (§5) in which the training data (for the machine) are sound-meaning pairs, and in which syntax is a hidden variable. This is a system which takes as a start the assumption that there is no external access to the internal states of a program such as Searle's. They use a limited category space in lieu of a universal grammar to control the search-space problem for the hypotheses the system generates, and we can assume that the substantive and formal principles can do the same task for the child in the manner described in §5. Therefore, the input to the room must be sound-meaning pairs in order for computation to take place inside the room, and syntax—more specifically parsing—is what happens inside.

Led this way, the system learns a fully interpretable grammar, of course with errors and approximations, but with the possibility of correcting them by exposure to further data. The crucial computationalist assumptions in their algorithm are that shorter, contiguous and less ambiguous strings are enter-

tained first, because the system must look at the powerset of alternatives to guarantee that the correct hypothesis is always among the candidates. Without these assumptions, we cannot assume that once hypothesis selection is down to a single candidate or very few candidates, we are done.

The results are too preliminary to be conclusive, but they point out principled directions for discerning the methodological and intrinsic problems of computing. I conclude that (a) Searle's Chinese Room is linguistically inadequate, and (b) it can be made consistent with bona fide computation, in which case the unduly pessimistic belief that a computational system cannot be made to face the same conditions for understanding as humans is not warranted. The key point is having access to semantics as an independent channel of intake and output, as assumed in the inverted-Y diagram of Figure 6.

In this setting, assuming an opaque computation by the invariants of CCG helps us narrow down the remedy when learning goes awry: the semantics of the invariants have no substantive constraint on their PADS. For example, composing *love'* and *hurt'* to get **B***love'hurt'* can only go wrong if we have the wrong assumptions about *love'* or *hurt'*. The semantics of **B** is invariant.

The opaqueness of the invariants and the transparency of the substantive assumptions (experiential knowledge) further reveal the nature of computation in CCG. It is a monad, where these processes are threaded rather than performed independently.

Chapter 10

Monadic computation by CCG

The possible landscape of substantive categories can be significantly reduced by considering the codetermination of syntax and semantics under a single fundamental assumption, adjacency. But it might seem excessive that CCG makes use of so many invariants as its combinatory base to do that (see Table 2 for a long list). The reason I have suggested is that factoring the combinations as such makes the grammatical process completely syntactic type-driven and transparent to the sources of types, to morphology, phonology and lexical semantics. Nothing needs to be remembered during a type-driven derivation. This seems to be a prerequisite to work towards understanding parsing as a reflex.

Nevertheless, one would expect in a purely applicative system that application as its primitive would stand out against all others. Recent analyses indeed suggest computationally distinguishing dependency and application in CCG. No constraint has been found necessary so far on the syntacticized combinators **B** and **S**, in controlling the projection of features of radically lexicalized types. (More accurately, all the earlier constraints on combinatory rules have been replaced by constraints on lexicalized syntactic types.) Combinatory dependencies always project all features.

Some constraints seem inescapable for application. It will follow that combinatory dependencies can be opaque processes whereas application must be transparent so that we can apply the constraints. These findings reveal the monadic aspects of CCG, suggesting that CCG's one-step computation is a two-stage process, as in monads.

Monads are quirky mathematical objects. They are in fact ubiquitous in everyday computing. For example, the famous Unix "pipe" (invented by Douglas McIlroy in the 60s) is represented as '|', and it threads a sequence of computations by chaining their input and output, which is now called the I/O monad. If n processes agree to take input and produce output in a standard way (called *streams*), we can chain them as $p_1 | p_2 | \dots | p_n$.

It is tempting to think of parsing as one long seamless pipe where every individual stage p_i is some parsing action (i.e. rule use, equivalently, for CCG, type use). However, this would imply that any intermediate process is opaque looking from the outside world. This is most likely not true, for

example we have catches of breath (or rest in signing), intonational phrasing, restarts, interjections, turn taking and giving (either voluntary or involuntary), etc. Some stages seem to be available for “repiping”, i.e. we have $p_1 | \dots | p_j | | p_{j+1} \dots | | \dots | | p_k | \dots | p_n$, rather than $p_1 | \dots | p_k | \dots | p_n$, where ‘|’ represents a joint in the pipe at which some properties must be transparent.

As the preceding preliminary discussion implies, I believe CCG as a theory has something to say about these “| | joints” where access is needed, and it has to do with the interaction of the seamless lexical projection of types onto surface phrases and satisfaction of constraints.⁹⁷ The applicative structure and dependencies seem to vary systematically in this regard.

1. Application

The asymmetry of simply projecting all the features in a combinatory rule and sometimes having to stop the projection in application is forced by the data. The following four instances among others corroborate the asymmetry of feature projection.

1.1. Reflexives

Consider again the simple control of grammar-lexicon division in CCG using the feature LEX, for “lexical”. Steedman and Baldridge (2011) argue that radically lexicalizing the reflexives forces a feature such as LEX. The category of the reflexive must look for a lexical verb:

$$(1) \text{ Mary} \quad \text{hurt} \quad \text{herself}$$

$$\frac{\frac{(S \backslash NP) / NP \quad (S \backslash NP) \backslash_{LEX} ((S \backslash NP) / NP)}{S \backslash NP}}{<}$$

CCG’s derivations are entirely syntactic type-driven therefore the syntactic type of *herself* must bear this feature as +LEX, as above, which we could also write as ‘\ell’ as before. We need this constraint to avoid reflexive interpretations of *herself* and *himself* in the example *John showed Mary herself/himself*. They are forced to a different analysis because, unlike true reflexives, they must take focal accent (Steedman, p.c.). Therefore application is subject to the following constraint:

- (2) $X/\ast\Lambda_1 Y \quad Y_{\Lambda_1} \quad \rightarrow \quad X_{\Lambda_2}$
 $Y_{\Lambda_1} \quad X \backslash \ast\Lambda_1 Y \quad \rightarrow \quad X_{\Lambda_2}$
 where Λ_i are variables for the value of LEX.
 $\Lambda_2 = \Lambda_1$ if Λ_1 is specified, $\Lambda_2 = -\text{LEX}$ otherwise.

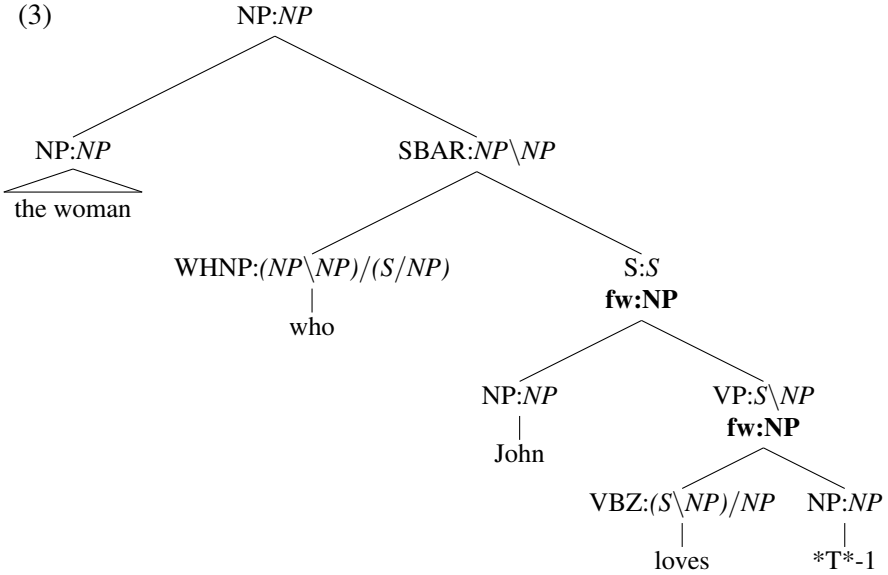
It would be projection if $\Lambda_2 = \Lambda_1$ necessarily. This seems to be the case for **B** and **S**. No special treatment has been reported for them in the literature. The earlier constraints on combinatory rules, for example those in Steedman (1985), have been replaced by the lexical control of slash modalities since Baldrige (2002). The only remaining constraint which has not yet been reformulated via modalities is Trechsel's (2000:630) stipulation on forward composition for Tzotzil, which is readily translatable to lexical restrictions as $S/\diamond NP$ for the unaccusative verbs and $S/\ast NP$ for the unergative verbs.

1.2. Supervised learning

The second example of projection asymmetry arises from a similar special treatment of application, for the purpose of learning the CCG categories from annotated data. Hockenmaier, Bierner and Baldrige (2004) report the following from the Penn Treebank:

```
(NP-SBJ (NP The woman))
  (SBAR (WHNP-1 who)
    (S (NP-SBJ John)
      (VP (VBZ loves)
        (NP (-NONE- *T*-1))
        (ADVP deeply))))))
```

They explain: "If a *T* trace is found and appears in complement position (as determined by the label of its maximal projection), a 'slash category' is passed up to the maximal projection of the sentence in which the trace occurs (here the S-node), hence signaling an incomplete constituent" Hockenmaier, Bierner and Baldrige (2004: 176). The passing of the slash category is shown in bold in the tree below, which is their CCG approximation for the same data. Its projection stops when the head daughter (e.g. *who* above) applies.



The implicit assumption is that the substring (*John loves *T**) is derived by CCG’s combinatory rules, viz. **B** here.

$$\begin{array}{c}
 (4) \quad \frac{\frac{John}{NP} \quad \frac{loves}{(S\backslash NP)/NP}}{S/(S\backslash NP)} \xrightarrow{T} \\
 \frac{\quad}{S/NP} \xrightarrow{B}
 \end{array}$$

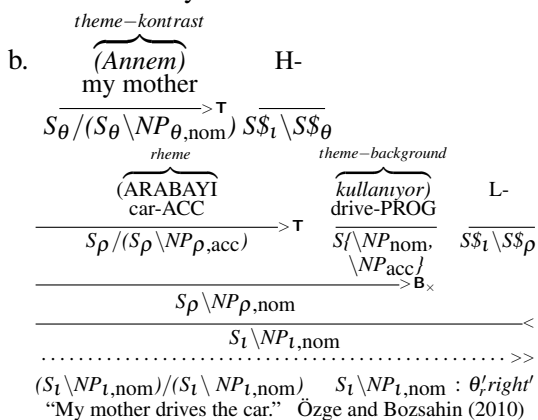
In this range this feature always projects, and the head closes off the projection by application, say with the category $(NP\backslash NP)/(S/NP)$ for *who*.

No special treatment of projection has been reported for combinator-like dependencies in wide-coverage parsing models, where large quantities of similarly annotated data are available for training; see e.g. Hockenmaier and Steedman (2007) for English, and Çakıcı (2008), Eryiğit, Nivre and Oflazer (2008) for Turkish.

1.3. Gapping and syntactic abstraction

The third example of a special treatment for application comes from information structure and focus projection. Steedman (2000b) has proposed a rule of decomposition for verb-medial gapping, which in effect does the triple duty

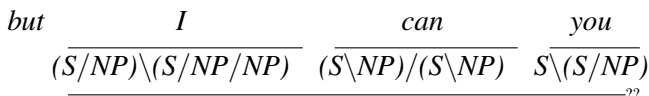
- (8) a. *Anne-n ne-yi kullan-ıyor?*
 mother-POSS.2s what-ACC use-IMPF
 ‘What does your mother drive?’



Thus the asymmetry between application and the combinatory rules such as **B** and **S** is maintained, and there is no reabstraction asymmetry between backward and forward application with respect to focus projection.

Before closing this section, I must note that forward abstraction is new evidence for syntactic abstraction. Backward abstraction might seem construction-specific, as it serves to only mediate gapping in verb-medial languages. It has its limits on constituency, for example the following example cannot have a rightward constituent to reveal, because Steedman’s (2000b) analysis crucially depends on both subject and the object to be backward type-raised (i.e. English is considered virtually VSO and lexically SVO to avoid anti-gapping and other bad effects), which cannot obtain a constituent in this example, as shown for the string *but i can you*.

- (9) *Yippeee, you can’t see me, but I can you.* Syd Barrett⁹⁸



However, both Steedman’s backward reveal rule and the forward one above do something crucial to syntactic types: they engender a mechanism for focus projection and its narrowing; see Özge and Bozsahin (2010) for some discussion. Not surprisingly, such processes are not sensitive to lexical material (but they are certainly dependent on syntactic properties such as being an argument versus predicate), hence their reverse process leading to abstrac-

tion in syntax seems justifiable. The essence of the rule seems necessary for incorporating that aspect of the syntactic process.

1.4. Morpholexical constraints

The fourth and final example of an exception to feature projection in application is reported for external sandhi, including Welsh soft mutation and English *wanna*-contraction. I have in mind Steedman's (2009) proposal to handle them in a unique way. Both processes require that we stop the sandhi, which is a finite-state rule in Steedman's formulation, under forward application, and always project this feature in combinatory rules.

The sandhi rule also seems relevant to backward application. In Turkish noun incorporation, where a morphologically unmarked preposed argument is incorporated into the adjacent verb, external sandhi is instigated by the SOV verb, i.e. by backward application: the incorporated noun is syllabified with the verb as external sandhi (10a–c), if phonological constraints on syllables are not violated as in (10d).⁹⁹ Brackets in these examples denote syllabic segments. (10e) shows that the morpholexical rule has limited applicability.

- | | | |
|---|----------------------------|---------|
| (10) a. <i>kitap okumak</i> | b. <i>taraf oldu</i> | Turkish |
| ..[ta][po].. | ..[ra][fol].. | |
| book read-INF | side be-PAST | |
| 'book reading' | 'be part of' | |
| c. <i>tuğla presledi</i> | d. <i>taraf tuttu</i> | |
| ..[lap][res].. | ..[raf][tut].. (no sandhi) | |
| brick pressed | side took | |
| 'brick-pressed' | 'took side' | |
| e. <i>tuğlayı presledi</i> | | |
| ..[yɪ][pɪ].. *[yɪp][res]... | | |
| brick-ACC pressed (no incorporation; no sandhi) | | |
| 'pressed the brick' | | |

In summary, there is something special about application in feature projection, whereas no special care is needed for combinatory dependencies. All manifestations of application, the forward and the backward varieties, seem prone to this asymmetry.

Application is also special theoretically because it cannot be conceived as a combinator itself *for* syntax or morphology. Although we can assume

$\mathbf{A} \stackrel{\text{def}}{=} \lambda f \lambda x. fx$, i.e. application as a lambda term without free variables, the juxtaposition fx is now unaccounted for if we do not take application as a primitive and employ \mathbf{A} in its stead. This would leave no room to syntacticize or morphologize \mathbf{A} ; we would need a primitive of concatenation or affixation.

2. Dependency

In contrast to application, dependency computation seems to be an opaque process. We can first have a look at some of the limited degrees of freedom afforded by this result, then exploit it to provide an efficient model of CCG's computation.

The invariants in Table 2 combine by application or combination. We can also think of them as unary type correspondences, which reveals their dependency encoding. I now write the semantics of correspondences as well to discuss the dependencies. Recall that, in CCG, dependencies manifest themselves in the predicate-argument structure, and syntacticization faithfully reflects them on syntactic types, as shown in Chapter 4.

I rewrite a fragment of Table 2 as a running example for this chapter. The first version is obtained by carrying the category of the nonhead term to the right of the arrow, which leaves the semantic head f as the input to deriving the right-hand side:

- (11) Semantics-driven encoding of dependencies:
- | | | | |
|--|---------------|---|------------------|
| $X/_*Y: f$ | \rightarrow | $X/Y: \lambda x. fx$ | $(>)$ |
| $X \backslash_* Y: f$ | \rightarrow | $X \backslash Y: \lambda x. fx$ | $(<)$ |
| $X/_\diamond Y: f$ | \rightarrow | $(X/_\diamond Z)/(Y/_\diamond Z): \lambda g \lambda x. f(gx)$ | $(> \mathbf{B})$ |
| $X \backslash_\diamond Y: f$ | \rightarrow | $(X \backslash_\diamond Z) \backslash (Y \backslash_\diamond Z): \lambda g \lambda x. f(gx)$ | $(< \mathbf{B})$ |
| $(X/_\diamond Y)/_\diamond Z: f$ | \rightarrow | $(X/_\diamond Z)/(Y/_\diamond Z): \lambda g \lambda x. fx(gx)$ | $(> \mathbf{S})$ |
| $(X \backslash_\diamond Y) \backslash_\diamond Z: f$ | \rightarrow | $(X \backslash_\diamond Z) \backslash (Y \backslash_\diamond Z): \lambda g \lambda x. fx(gx)$ | $(< \mathbf{S})$ |

Here I follow the standard practice in computational linguistics, which dates back to Partee and Rooth (1983), that categories enter the lexical assignments at their “lowest type”. In (11), they are the left-hand sides of the arrow. For example, $\alpha := (X/_\diamond Z)/(Y/_\diamond Z)$ is a higher-type homonym of $\alpha := X/_\diamond Y$ in Partee and Rooth's sense. A right arrow for a unary correspondence must be interpreted in the current chapter with this assumption in mind.

The correspondences in (11) follow from adjacency because, if we have the configuration $A B \Rightarrow C$, we can also get it with $A \rightarrow C/B$ and $B \rightarrow C \backslash A$.

The dependencies in (11) arise from a semantic (head-driven) strategy of translating the CCG rules in Table 2 to the homonyms of their head function. For example, in ($< \mathbf{B}$), the head function f is phonologically after g because $Y \setminus_{\circ} Z: g \ X \setminus_{\circ} Y: f \rightarrow X \setminus_{\circ} Z$. We kept f on the left-hand side of the correspondences in (11), which is the head of the dependency in $\mathbf{B}fg$.

In contrast, the homonyms of the lower types in (12) are motivated by phonology because it is always the phonologically first category of a combinatory rule that is mapped to a homonym, which is guaranteed by adjacency. It is clear from these correspondences that the first two rows of the phonological strategy (12) and the semantic strategy (11) add nothing informative to the set of categories which are at the parser's disposal, with the exception of the phonological translation of ($<$) in (12), which looks suspiciously like forward type raising (more on this later).¹⁰⁰ The last four rows of each strategy are informative syntactically.

(12) Phonology-driven encoding of dependencies:

$$\begin{array}{lll}
 X/_*Y: f & \rightarrow & X/Y: \lambda x.fx & (>) \\
 Y: a & \rightarrow & X/(X \setminus_* Y): \lambda f.f a & (<) \\
 X/_\circ Y: f & \rightarrow & (X/_\circ Z)/(Y/_\circ Z): \lambda g \lambda x.f(gx) & (> \mathbf{B}) \\
 Y \setminus_{\circ} Z: g & \rightarrow & (X \setminus_{\circ} Z)/(X \setminus_{\circ} Y): \lambda f \lambda x.f(gx) & (< \mathbf{B}) \\
 (X/_\circ Y)/_\circ Z: f & \rightarrow & (X/_\circ Z)/(Y/_\circ Z): \lambda g \lambda x.f x(gx) & (> \mathbf{S}) \\
 Y \setminus_{\circ} Z: g & \rightarrow & (X \setminus_{\circ} Z)/((X \setminus_{\circ} Y) \setminus_{\circ} Z): \lambda f \lambda x.f x(gx) & (< \mathbf{S})
 \end{array}$$

Current thinking in linguistics is that phonological cues tune in earlier than semantics, and they are predictive. This view favors a model of CCG parsing which uses (12) for dependencies, rather than the binary rules of Table 2, or the semantically-motivated (11).¹⁰¹

The learning of a syntactic category is the crucial part of acquiring a grammar, which is a *hidden variable* problem (Zettlemoyer and Collins 2005), where the input is a pairing of a phonological form and an assumption about its meaning (the PADS), and the syntactic type is the hidden variable. (12) suggests that the learner in the parse-to-learn paradigm first has a grip on the type homonyms of the prefixes of a string to associate right or wrong meanings with parts of it. Thus the string must be available as a structured domain for analysis, so that we can hypothesize about the syntactic types from the beginning to the end in a sequential fashion.

All CCG learners operate in the parse-to-learn paradigm, for example Villavicencio (2002), Bos et al. (2004), Zettlemoyer and Collins (2005), Çöltekin and Bozsahin (2007), Steedman and Hockenmaier (2007), Clark and

Curran (2007). We have seen the basic idea at work in §9.5. They can be made to work with (12) to implement the phonological-cues-first idea, provided that we can thread application and combinatory dependencies to achieve the pipeline effect of (12), which we will do in the next two sections. Presumably such parsing models will be easier to train on phonological cues.

The last four rows of (11) and (12) are all nonredundant. The list in Table 6 shows that nonredundancy holds even in the presence of crossing modalities and powers. The list is a phonological encoding of Table 2.

The phonological encoding may be the most natural monadic computation by CCG because through it all dependencies begin to look forward in the string. Notice the main slashes of the higher-type homonyms in Table 6, which are in the right-hand sides of the arrow. This encoding's relation to adjacency is evident (recall that we have no phonologically null type assignments), and this takes us to sequencers in combinatory theory.

Whether we keep the dependencies separate or phonologically or semantically encode them depends on what use we put them to. In all cases, a freely operating \mathbf{T} is absent, and this also directly relates to sequencing. This is one aspect which stands out in a monadic interpretation of CCG.

3. Sequencers

The implication of the results so far is that dependency projection in a parsing configuration can be an opaque process. Although application requires its ingredients to be visible so that idiosyncratic constraints can be imposed on it, no such visibility seems required for dependencies. In other words, dependencies can be shunted into a sequencer, whereas application cannot.

The combinatory equivalent of a sequencer is Curry and Feys's (1958) composite product, defined as $X \cdot Y \equiv \mathbf{B}XY$. As they proved, it is equivalent to a sequencing of X and Y , where $X \cdot Y$ first performs X , then Y , on a sequence of arguments. For sequencing to work X must be a *regular* combinator, i.e. it must not change the order of its first argument. Therefore, \mathbf{T} cannot be a sequencer, because $\mathbf{T} \stackrel{\text{def}}{=} \lambda x \lambda y. yx$.

Its import for monadic CCG is that \mathbf{T} cannot be part of a monad that contains Table 6. Take $\mathbf{T}\mathbf{B}fgh$, which reduces to $f\mathbf{B}gh$, and there is no possibility of reaching a normal form. This would abruptly stop the monad after its first step. The relevance of this result to efficient computation will be evident in the next section, but first let us look at what more is at stake with \mathbf{T} .

Table 6. Phonology-driven encoding of monadic dependencies (cf. Table 2).

$X/_\diamond Y$	\rightarrow	$(X/_\diamond Z)/(Y/_\diamond Z)$	$> \mathbf{B}$
$Y \backslash_\diamond Z$	\rightarrow	$(X \backslash_\diamond Z)/(X \backslash_\diamond Y)$	$< \mathbf{B}$
$X \backslash_\times Y$	\rightarrow	$(X \backslash_\times Z)/(Y \backslash_\times Z)$	$> \mathbf{B}_\times$
$Y \backslash_\times Z$	\rightarrow	$(X \backslash_\times Z)/(X \backslash_\times Y)$	$< \mathbf{B}_\times$
$X/_\diamond Y$	\rightarrow	$((X/_\diamond Z) W)/((Y/_\diamond Z) W)$	$> \mathbf{B}^2$
$(Y \backslash_\diamond Z) W$	\rightarrow	$((X \backslash_\diamond Z) W)/(X \backslash_\diamond Y)$	$< \mathbf{B}^2$
$X \backslash_\times Y$	\rightarrow	$((X \backslash_\times Z) W)/((Y \backslash_\times Z) W)$	$> \mathbf{B}_\times^2$
$(Y \backslash_\times Z) W$	\rightarrow	$((X \backslash_\times Z) W)/(X \backslash_\times Y)$	$< \mathbf{B}_\times^2$
$(X/_\diamond Y) \backslash_\diamond Z$	\rightarrow	$(X/_\diamond Z)/(Y/_\diamond Z)$	$> \mathbf{S}$
$Y \backslash_\diamond Z$	\rightarrow	$(X \backslash_\diamond Z)/((X \backslash_\diamond Y) \backslash_\diamond Z)$	$< \mathbf{S}$
$(X \backslash_\times Y) \backslash_\times Z$	\rightarrow	$(X \backslash_\times Z)/(Y \backslash_\times Z)$	$> \mathbf{S}_\times$
$Y \backslash_\times Z$	\rightarrow	$(X \backslash_\times Z)/((X \backslash_\times Y) \backslash_\times Z)$	$< \mathbf{S}_\times$
$(X/_\diamond Y) Z$	\rightarrow	$((X/_\diamond W) Z)/((Y/_\diamond W) Z)$	$> \mathbf{S}''$
$(Y \backslash_\diamond W) Z$	\rightarrow	$((X \backslash_\diamond W) Z)/((X \backslash_\diamond Y) Z)$	$< \mathbf{S}''$
$(X \backslash_\times Y) Z$	\rightarrow	$((X \backslash_\times W) Z)/((Y \backslash_\times W) Z)$	$> \mathbf{S}_\times''$
$(Y \backslash_\times W) Z$	\rightarrow	$((X \backslash_\times W) Z)/((X \backslash_\times Y) Z)$	$< \mathbf{S}_\times''$
$X/_\diamond (Y Z)$	\rightarrow	$(X/_\diamond (W Z))/(Y/_\diamond W)$	$> \mathbf{O}$
$Y \backslash_\diamond W$	\rightarrow	$(X \backslash_\diamond (W Z))/(X \backslash_\diamond (Y Z))$	$< \mathbf{O}$
$X \backslash_\times (Y Z)$	\rightarrow	$(X \backslash_\times (W Z))/(Y \backslash_\times W)$	$> \mathbf{O}_\times$
$Y \backslash_\times W$	\rightarrow	$(X \backslash_\times (W Z))/(X \backslash_\times (Y Z))$	$< \mathbf{O}_\times$

A set-theoretic formulation makes it easier to see that \mathbf{T} is a monoid itself in a system of binary combination. Recall Lambek's (1988) definition of type-raised categories, for example $S/(S \backslash N)$ as $N,^S$ where he also noted $(N^S)^S = N.^S$ Given three functions f, g and h in space N^S we have $f \circ (g \circ h) \Leftrightarrow (f \circ g) \circ h$, where \circ is binary composition. The identity element of the monoid is $\lambda f \lambda x. x f$ for $f \in N.^S$ The meaning of this result for our present concern is that the monoid adds further bracketings of the same string, therefore it is not part of the core system, and it must be controlled lexically to stop the overgeneration of surface constituency.

The idiosyncrasy of the syntacticized \mathbf{T} (type raising) and the argument-hood requirement on its application (it applies to argument types to yield

functions from functions that require such kinds of arguments) also suggest that type raising must continue to operate in a lexically constrained manner. The alternative, which is to incorporate **T** as $X/(X \setminus Y)$ in Table 6, leads to nonmild-context-freeness, as Hoffman (1993) showed. Similarly, the backward variety $X \setminus (X/Y)$ must be avoided. Thus **T**'s problems with sequencing are confounded by its uncontrollable power of producing categories indefinitely without advancing the computation, if let to operate freely.

Although lazy evaluation can be called in to force a freely operating **T** to terminate in a parsing configuration, the problem with **T** is not only effective computability or disruption of sequencing. An implication of Curry and Feys's (1958) results is that **T** is not dependency-encoding but dependency-preserving: $\mathbf{T} = \mathbf{CI}$. Thus $\mathbf{T}fa = \mathbf{CI}fa = \mathbf{I}af = af$. The combinator **C** encodes a dependency between the head x and its arguments y and z in $\mathbf{C}xyz$, but this is neutralized by **I** in $\mathbf{CI}fa$. There is no lexical resource in **I** to encode a dependency, which by definition needs a head.

It is therefore not surprising that **T** must be part of the lexicalized grammar, either built into the lexical categories or operated as a lexical (unary) rule, and it must not be in the set of common dependencies that feed into application. And without **T**'s disruption of sequencing, other combinators manifest monadic computation with respect to application.

4. The CCG monad

A *monad* in Category Theory is a triple (M, η, μ) where M is the type constructor, η is the function to inject values into the monad by monadic type construction, and μ is the function that threads the computation within the monad. All monads can be characterized as below, where we need to fix η and μ to get a particular computation (' \rightarrow ' here represents a function's type).

$$(13) \quad \begin{aligned} \eta &: \kappa \rightarrow M\kappa \\ \mu &: M\kappa \rightarrow (\kappa \rightarrow M\rho) \rightarrow M\rho \end{aligned}$$

In our case, μ threads dependency and application. The computation engendered by CCG is known as the *reader monad* or the *parser monad* in programming languages (see Hutton 2007), which we can define as follows:

$$(14) \quad \begin{aligned} \text{Let } \text{CCG-M} &= (M, \eta, \mu), \text{ such that} \\ \eta c &= \lambda x.(c, x) : M\kappa, \text{ for } c, x : \kappa, \\ \mu &= \mathbf{S}. \text{ Thus } \mu a(dU) = \mathbf{S}a(dU) = \end{aligned}$$

$$\lambda x.ax(dUx): M\kappa \rightarrow (\kappa \rightarrow M\rho) \rightarrow M\rho,$$

where d is the dependency function, U is the set of dependencies in Table 6, and a is application.

Monadic computation has been known in computing since Wadler (1990) and Moggi (1991). The monadic nature of function application was pointed out earlier by Shan (2001), who was also among the first to point out the relevance of monads to natural language computation. Later, Barker and Pryor (2010) have shown that Jacobson's variable-free semantics constitutes a reader monad as well (her g and z ; see §6.3).

The inner workings of the CCG monad in (14) can be described as follows. The $M\kappa$ types will be pierced into κ types in the monad to do its computation; note the function type of d , viz. $\kappa \rightarrow M\rho$. The output of the monad is of type $M\rho$, which is the sequence containing a singleton result category because of the uniqueness of the dependencies in Table 6. The abstraction ηc injects an ordered pair of categories (of type $M\kappa$) into the monad.¹⁰² The result of the process d is a monadic homonym of the left component of x in U , depending on the right component of x . In simpler terms, if a left-hand side in Table 6 matches a left component of the input in the monad, and if the right component in the input matches the domain type of the homonym of the left component, then the homonym and the right component is returned as a pair. Failure is reported as \star .

The result of the monad is the result of process a , which can be forward application of CCG, backward application, a binary use of a common dependency in U (Table 6), or failure, reported as \perp . Function a applies the result of (dUx) to the input in x . Thus the knowledge of common dependencies is kept in the monad as its internal affairs.

Some examples can clarify the mechanism. Assume that we inject into the monad the sequence $(S \setminus NP, S \setminus_{\diamond} S)$. The evaluation of the process (dUx) yields $((S \setminus NP) / (S \setminus_{\diamond} S), S \setminus_{\diamond} S)$ by $(< \mathbf{B})$. Function a becomes the forward application of $(S \setminus NP) / (S \setminus_{\diamond} S)$ to $S \setminus_{\diamond} S$. If the input were $(NP, S \setminus NP)$, the process (dUx) would return \star because no common dependency manifests a leftward nonfunctor type such as NP . The process $(ax\star)$ then becomes backward application, where x is the pair $(NP, S \setminus NP)$.

Consider now the input $(S / NP, NP)$. Process (dUx) returns \star because no dependency in U can match NP as the domain type, and $(ax\star)$ becomes forward application. The monad would report failure (\perp) for the ordered pair $(N, (N \setminus N) / NP)$ because neither d nor a succeeds.

To avoid confusion of monadic CCG derivations with standard CCG derivations which decorate rules on the right-hand side of a line of derivation, I index the monadic derivations on the left-hand side of a line, and write the monadic combinator that led to the successful application of (14):

(15) a. *Wittgenstein loathed and Kafka adored mentors.*

$$\frac{\overline{S/(S\backslash NP)} \quad \overline{(S\backslash NP)/NP}}{>\mathbf{B} \quad \overline{S/NP}}$$

b. *Articles which I will file without reading*

$$\frac{\overline{VP/NP} \quad \overline{(VP\backslash VP)/C_{\text{ing}}} \quad \overline{C_{\text{ing}}/NP}}{>\mathbf{B} \quad \overline{(VP\backslash VP)/NP}} \\ <\mathbf{s}_x \quad \overline{VP/NP}$$

Example (15a)'s inner workings can be fleshed out as a two-stage process of one-step computation by CCG-M, as in (16).

(16)

$$\frac{\overline{Wittgenstein} \quad \overline{loathed}}{\overline{S/(S\backslash NP)} \quad \overline{(S\backslash NP)/NP}} \\ \overline{\overline{(S/NP)/((S\backslash NP)/NP)}^1} \\ \overline{\overline{\dots\dots\dots}^2} \\ \overline{S/NP}$$

It first applies ($>\mathbf{B}$) of Table 6 to $S/(S\backslash NP)$ to get $(S/NP)/((S\backslash NP)/NP)$ by process d , then forward-applies it to the category of *loathed* to yield S/NP by process a . This is binary \mathbf{B} as a two-stage process, shown in dotted lines above.

There is no unary application of the combinatory dependencies by the monad; the unary dependencies of process d must always thread through process a . If allowed unary application would be a dangerous practice because we know that unary \mathbf{B} must not operate freely in syntax. I repeat the relevant examples, where (17b) attempts unary \mathbf{B} :

(17) a. *I think that Wittgenstein might have liked Kafka*

b. **I think Wittgenstein that might have liked Kafka*

$$\frac{\overline{VP/S'} \quad \overline{S'/S_{\text{fin}}} \quad \overline{S_{\text{fin}}}}{\overline{NP} \quad \overline{S'/S_{\text{fin}}} \quad \overline{S_{\text{fin}}\backslash NP}} \\ \overline{\overline{(S'\backslash NP)/(S_{\text{fin}}\backslash NP)}^{\mathbf{B}}} \\ \overline{S'\backslash NP}$$

The only monadic dependency that can force (14) to combine ‘that’ with ‘might have liked Kafka’ is ($> \mathbf{B}_\times$) below (repeated from Table 6):

- (18) a. $X \backslash_\times Y \rightarrow (X \backslash_\times Z) / (Y \backslash_\times Z)$ ($> \mathbf{B}_\times$)
 b. *that might have liked Kafka*

$$\frac{\frac{S' \backslash_\times S_{\text{fin}}}{> \mathbf{B}_\times} \quad S_{\text{fin}} \backslash NP}{S' \backslash_\times NP}$$

However, this combination requires the lexical assumption $S' \backslash_\times S_{\text{fin}}$ for the complementizer, which is empirically inadequate: we cannot derive the following fragment.

- (19) *the field I think that Kafka liked*

$$\frac{S' \backslash_\times S_{\text{fin}} \quad S_{\text{fin}} / NP}{??}$$

We can radically lexicalize the contrast in (18) and (19) in the category of *that* without further assumption, which must be $S' / \diamond S_{\text{fin}}$, as standardly assumed in CCG:

- (20) *that Kafka liked*

$$\frac{S' / \diamond S_{\text{fin}} \quad S_{\text{fin}} / NP}{> \mathbf{B} \quad S' / NP}$$

Likewise, a freely operating unary \mathbf{S} is dangerous. Consider the Welsh examples again, from Awbery (1976: 39). Although the category $(S/S') / NP$ is sound for the complement-taking verbs (21a), the word order instigated by a unary \mathbf{S} from this category is ungrammatical (21b). Welsh is strictly VSO, and the verb must avoid unary \mathbf{S} .

- (21) a. *Dymunai Wyn i Ifor ddarllen llfyr.* Welsh
 Wanted Wyn for Ifor reading (a) book

$$\frac{(S/S') / NP \quad NP \quad S'}{S'}$$

 ‘Wyn wanted Ifor to read a book.’
 b. **Dymunai ddarllen llfyr Ifor*

$$\frac{(S/S') / NP \quad S' / NP \quad NP}{> \mathbf{S} \quad (S / NP) / (S' / NP)}$$

Thus we can assume dependencies to operate freely *within* the monad, where they only serve as an input to binary juxtaposition. The same can be said about combinators, which no longer decide rule choice and simply project dependency encoding.

Combinatory modalities encoded in slashes continue to do nonredundant work in the syntacticization of monadic dependencies. For example, given the sequence $(S \setminus_{\diamond} NP, S \setminus_{\star} S)$, process (dUx) can only produce $((S \setminus_{\diamond} NP) / (S \setminus_{\diamond} S), S \setminus_{\star} S)$, not $((S \setminus_{\diamond} NP) / (S \setminus_{\star} S), S \setminus_{\star} S)$; cf. the definition of $(< \mathbf{B})$ in Table 6. The process a of the monad fails because the application $(S \setminus_{\diamond} NP) / (S \setminus_{\diamond} S) \setminus_{\star} S$ fails. Thus the following expected behavior is respected:

$$(22) \quad * \textit{player} \quad \textit{that} \quad \textit{shoots} \quad \textit{and} \quad \textit{he misses} \quad \textit{Baldrige} \quad (2002)$$

$$\frac{(N \setminus_{\diamond} N) / (S \setminus NP) \quad S \setminus_{\diamond} NP \quad (S \setminus_{\star} S) /_{\star} S \quad S}{\frac{S \setminus_{\star} S}{* < \mathbf{B} \quad * S \setminus_{\diamond} NP}} >$$

Likewise, the configuration $(S \setminus_{\star} S, S \setminus_{\diamond} S)$ fails to make use of the dependency encoded by $(< \mathbf{B})$ because no left member of Table 6 can match $S \setminus_{\star} S$. Therefore the relevant monadic relation among the slash modalities is that the input to the dependency must be a supertype, as before.

5. Radical lexicalization revisited

No combinatory dependency in Table 6 relies on or introduces a star modality. This move makes all the slash modalities truly lexical because we no longer need to write the sole combinatory rule of monadic combination, (14), with modalities. Modalities only encode the differential lexical syntacticizations of semantic dependencies, for example harmonic versus disharmonic composition, or no composition.

It is not surprising that the star modality never appears in the repertoire of common dependencies in Table 6: it does not encode a dependency at all because it cannot involve a syntactic combinator. This is explicit in a monadic CCG.

The parsing configurations for imposing the special restrictions on application, some of which are listed in §1, are uniquely identifiable in a monadic CCG as $(ax\star)$. This is the condition in which all dependencies fail, and juxtaposition is the only possibility left for combining. This is an important source of information for the oracle, because it needs a limited window of parsing contexts, in addition to individual word statistics and some transitional probabilities, to be able to decipher the relevance of special constraints. We can refine this configuration as $(a(X/Y, Y)\star)$ and $(a(Y, X \setminus Y)\star)$, to distinguish

the unique conditions in which forward and backward application are possible. These slashes do not need modalities (we can say they bear the least restrictive type ‘.’) because application is already implicated by \star . Therefore, there is only one primitive of combination, viz. the function μ in the monad (14).

The cryptic notation of monadic CCG is for a good cause. It manifests the same dependency as the ternary, binary and unary equivalents in the standard notation, but embodies (i) phonological precedence, (ii) semantic headness and (iii) the single slash of combination (the always-forward-looking main slash without modalities), all in the left-edge of a combination. A comparison of the alternatives for backward crossing composition below show that this is indeed the case.

$$\begin{array}{llll}
 (23) & Y/\times Z: g & X\backslash\times Y: f & Z: a \rightarrow X: f(ga) & \text{(ternary)} \\
 & Y/\times Z: g & X\backslash\times Y: f & \rightarrow X/\times Z: \lambda x.f(gx) & \text{(binary)} \\
 & Y/\times Z: g & & \rightarrow (X/\times Z)/(X\backslash\times Y) & \text{(unary)} \\
 & & & : \lambda f\lambda x.f(gx) & \\
 & (X/\times Z)/(X\backslash\times Y) & X\backslash\times Y: f & \rightarrow X/\times Z: \lambda x.f(gx) & \text{(monadic)} \\
 & : \lambda f\lambda x.f(gx) & & &
 \end{array}$$

For example, the directionality of all functions and arguments are preserved. Compare the ternary version with the monadic one. Z 's directionality is forward, Y 's directionality is backward, and X as the result is not associated with a directionality. All of these are maintained in the monadic version. The head functor, f , is anticipated in the monadic variety from the phonologically earlier string.

We can assume for the benefit of the computationalist treatments of language acquisition that all three aspects (i–iii) above are conveniently located in the first category, and that the monadic version can be compiled out from the standardly assumed binary version.¹⁰³

All syntactic dependencies are forward-looking in Table 6, as one would expect from a phonology-driven base for a strict competence grammar. All of them arise from the semantics of combinators noted on the far right, and all of them are forward juxtapositions, as expected from the syntactic correlate of the phonological attachment.

Finally, note that in the CCG monad of (14) the dependency computation by process d terminates in $O(k)$ time through a naive search algorithm, where k is the size of the set (a constant) in Table 6, if the set does not contain \mathbf{Y} , \mathbf{K} or \mathbf{I} . No data have been forthcoming for a syntacticization of these combinators.

6. Monadic results and CCG

As is evident from Table 6, the monadic grammar described in this chapter is functionally equivalent to CCG. It avoids unary use of combinators, and it is forced to keep **T** out of the monad, which is similar to CCG's substantive constraints on type raising. So what good is a monadic CCG? The monadic perspective imports several results to CCG, as follows.

The asymmetry of application and dependency in feature projection suggests potentially different treatments of these aspects. The dependencies must always project (they are the opaque part of the monad) whereas application can “close off” a projection for a specific feature. These are different kinds of parsing actions. This is explicit in a monad. The distinction seems crucial for interfacing parsers with other components of language processing, for example with inference systems, learning systems or with systems of discourse and pragmatics.

Application in CCG can be reduced to a single primitive of parsing action, viz. juxtaposition, as originally intended by Schönfinkel (1920) for combinators almost a century ago. This potential simplification, at least in theory, shows that CCG adds no auxiliary assumptions to engendering constituency, dependency and structure from order alone.

The binary syntax of CCG follows not only from empirical concerns over unary and ternary **B** and **S**, but also from juxtaposition itself. The reason is as follows. If CCG is indeed monadic, then dependency projection must be internal to the monad, therefore the dependencies that are shunted into the monad cannot differ in arity. Nor can they combine by themselves if their output must feed into a primitive. The only noncombining unary-operating combinator is **T**, which does not instigate a dependency, and which is itself a monoid with severe limitations. We have good mathematical, computational and linguistic reasons to leave it out of the syntactic combinators. The last two aspects have been known for quite some time (e.g. Steedman 1987, 2000a, Komagata 1997, Hoffman 1993, Eisner 1996). The monadic perspective relates the two aspects to the mathematics of sequencing.

We would not expect to see in natural language syntax the kinds of dependencies Smullyan (1985) attributed to his admittedly odd combinator birds: Finch, Owl, Queer Bird, Quirky Bird, Robin, Turing Bird and Vireo. Like

Thrush (**T**), they are irregular combinators hence not sequencers. (A combinator is *regular* if it does not change the order of its first argument.) Although we can conceive a monadic organization of **T** (which is a monoid) and the set of common dependencies in Table 6 (which is a monad), in the form of a *layered* monad (Filinski 1999), which is to say that **T** can prepare input to the monad in (14), or apply unarily to its result, there is no indication that other spoilers add up to a monad with that layered monad. Considering the fact that **T** is part of the minimal apparatus **BTS** which captures the unorthodox but fully interpretable constituencies, these kind of dependencies do not seem to be relevant to natural language computation.

Hoyt and Baldridge (2008) derive the binary rules of CCG from unary **B**, **S** and **T**. Monadic CCG suggests that going the other way, i.e. deriving the unary dependencies from combinatory rules to factorize dependency and application is revealing too, theoretically and empirically. For one thing, we must assume the “derivational oracle” of the same derived meanings to be the speech data themselves, i.e. tones, tunes, stress and pitch accents. The reasons are as follows.

Hoyt and Baldridge introduce on the formal side *inert* slashes to do normal form parsing, and to derive the CCG rules from unary combinators. They also need a structural postulate in lieu of a switch to do normal-form versus left-branching parsing. We would like to be able to assume that the data contains the right source for disambiguation.

This is a forced assumption in a monadic CCG. Consider an inert-slash formulation of a homonym of ($> \mathbf{B}$), taken from Hoyt and Baldridge:

$$(24) X/_\circ Y: f \rightarrow (X/_\circ^! Z)/(Y/_\circ^! Z): \lambda g \lambda x. f(gx)$$

Monadic dependencies are opaque inputs to function application, thus we would be forced by this rule to introduce the antecedent-government semantics of ‘!’, equivalently the ‘-LEX’ restriction on the slash following Steedman’s (1996b) semantics for it adopted by Hoyt and Baldridge. Now we have to introduce ‘!’ to all ($> \mathbf{B}$) homonyms of $X/_\circ Y$. The rule above would force the monadic homonym to *always* require Z to be nonlexical, hence this constraint would not necessarily arise from a lexical category as one would expect.

An inert slash is fine as an option in the lexicon, but its introduction by a monadic homonym is problematic. Using the inert slash to avoid spurious derivations must then be reconsidered in light of the richness in the data. Take $[[John\ likes]\ Mary]$ and $[John\ [likes\ Mary]]$. Normal-form parsers (for

example that described in Eisner 1996) would eliminate the first alternative unless it is in a substring such as *the lady I believe John likes*, but these derivations are not spurious until we know the context and intonation in which the utterance took place:

- (25) a. *Who does John like?*
 b. *Why does John avoid Mary?*
 a'. [*John likes*] *Mary*]
 b'. [*John* [*likes Mary*]]

Both bracketings in (25a'–b') are possible analyses as an answer to the first question. The first analysis is spurious for the second question. We can assume, following Steedman (2000a), that the intermediate phrase boundary tones in an answer to the second question would not allow the bracketing [*John likes*] anyway. They have their own syntactic type, and in a modalized CCG, these types could not be composed over; they are $S\$ \setminus_x S\$$.

It is mainly the text data, i.e. information loss, that should make us wary of spurious derivations. Then we are left with spurious derivations within an intermediate phrase to worry about, which are related to focus projection and quantifier scope as well (Prevost 1995, Komagata 1999, Steedman 1999). Therefore it might be preferable to filter them out only after they are engendered, as done by Vijay-Shanker and Weir (1990), rather than avoid them syntactically as Eisner (1996) does. We can be aware of these consequences when we derive the unary dependencies from binary ones, as in the monad.

Monadic grammar might also suggest a principled way to answer the following question: why is the only kind of syntactic abstraction related to the primitive (juxtaposition)? Why can't we have **B**-abstraction, say in the reverse direction of ($> \mathbf{B}$) in the universal syntax of CCG?

Nothing can undo or redo a derivation to the extent of reconsidering the projection of lexically specified semantic dependencies. This is what combinators do on the PADS side of the words in the lexicon, which is directly reflected on their syntactic type. This follows not from a principle or a stipulation, but from the inherent asymmetry of sequencing the processes of dependency projection and application. The asymmetry is explicit in a monadic grammar.

Finally, let us reconsider the computational problem of language acquisition (§9.5) from a monadic perspective.

CCG has concentrated so far on head-driven approaches to the acquisition of categories (e.g. Niv 1994, Villavicencio 2002, Zettlemoyer and

Collins 2005, Çöltekin and Bozsahin 2007). A left-dependent monadic grammar forces us to follow a phonological line exemplified in Table 6. Semantics helps to narrow down the hypothesis space, thus the proposed sketch assumes—following Steedman and Hockenmaier (2007) and Chomsky—that the root of the problem is grammatical bootstrapping rather than syntactic or semantic bootstrapping.

The idea of the sketch coincides with a remark by Chomsky, quoted as personal communication by Hornstein (1995):

The basic point seems to me simple. If a child hears English, they [*sic*] pick up the phonetics pretty quickly (in fact, it now turns out that many subtle distinctions are being made, in language specific ways, as early as six months). The perceptual apparatus just tunes in. But if you observe what people are doing with language, it is subject to so many interpretations that you get only vague cues about LF.

This is entirely consistent with computationalist language acquisition outlined in §9.5. Recall that in that way of thinking the so-called universal grammar, in present terms the invariants and the constitutive principles for categories, sets up the prior probabilities of day one. The child's confidence in a category for strings, a posterior probability, is updated by a learning scenario where possible derivations set the stage for the likelihood, fostered by priors updated by experience.

Monadic grammar's contribution to this process is making phonology the driving force, where the left edge is not necessarily the semantic head as can be seen in Table 6. In any belief update on categories, we first see the left edge of a derivation, which is temporally the first part of a combinatory context. But the whole process cannot depend on phonology, because a left-dependent has many potential results the resolution of which depends on the right-dependent (i.e. expectation), and on the child's beliefs about the categories, that is, on her lexicon.

The potential result can be ambiguous only if the child's assumptions about the strings are ambiguous, because monadic application is always functional, given two adjacent types. Therefore any source of ambiguity must emanate from lexical types, or forced on the system from the outside, to be handled by an oracle. The process in between is an algorithm, which I wrote as the monad CCG-M.

Chapter 11

Conclusion

We started with Schönfinkel, then moved to Chomsky, Curry, Lambek, Geach, Bach, Montague and Gazdar, and reached Steedman, Szabolcsi and Jacobson as the progenitors of rule-to-rule semantics in applicative syntax, be it natural or formal. We ended with Schönfinkel through monadic grammar, where there is only one rule of syntactic combination. We dealt mostly with combinatory matters and only occasionally with set-theoretic ones, which deserve a book of their own.

Schönfinkel's ingenious method of variable elimination reveals adjacency as the sole basis of semantics, which, by virtue of Steedman's syntacticization, is also the sole basis of syntax. Ades and Steedman's (1982) Adjacency Corollary is an independent discovery of syntactic interpretability by juxtaposition alone, where they provide the first syntacticization of **B**. Geach (1972) is perhaps the first mention of combinators in syntax, where he follows Quine rather than Schönfinkel in variable elimination. I will come back to this point shortly.

It seems clear that Steedman's program is not the elimination of variables but keeping adjacency as the only base for syntax, which exports direct and immediate interpretability to constituents. His choice of LF as a level of representation attempts to resolve some unsettling issues of immediate interpretability, namely that of pronouns and scope variation, precisely because their semantics do not seem to arise according to Steedman from semantics of order alone. The variable-free semantics of Jacobson appears to have a different agenda, where adjacency in surface structure can be compromised, such as by a potential consideration of wrap for the benefit of (almost) interpretation-ready semantics, given some model-stage storage for binding and scope. Here adjacency is a cherished assumption but not a must. The key issues in the debate appear to be the impact of type-shifting rules on computational efficiency, the predictive force of positing or not positing an LF, and the unsettled nature of intermediate scope readings and others that fall between the cracks in scope-taking.

Combinators are not alone in persisting that only order leads to structure in cognitive science. Elman's (1990) simple recurrent networks take the notion of time out of input representations, and predict its structure from sequential

representations. The lesson we learned from such kind of connectionism is not only that symbols need some representational support, but that the inherent asymmetry of sequential representations can change the way we look at cognitive problems. The same can be said about combinators.

Schönfinkel's desire to find the foundations of mathematical logic has become a linguistic theory in CCG in which the only primitive is his Schönfinkelization of argument-taking objects by which not only arguments but functions can be passed on as values. Curry's similar aspirations have given us functional programming par excellence.

Curry's brief foray into linguistics, Curry (1961), suggested another way of handling natural language syntax-semantics where a combinatory calculus drives the logical aspects (he had called it the tectogrammatical level). A separate syntactic calculus (his phenogrammatical level) works on the surface structure engendered by words and phrases.¹⁰⁴ This line of research culminated in what is known as Applicative Grammar (Shaumyan 1977, 1987) and Convergent Grammar (Pollard 2008a).

It is interesting that Chomsky (1961) was in the minority in the famous symposium that was held in New York in April 14–15 1960, which also included papers on mathematics and language by Curry, Halle, Harary, Hockett, Jakobson, Lambek, Mandelbrot, Putnam, Quine and Yngve, among others. He suggested contra Curry and many others in the meeting—Lambek excluded—a unification of grammatical description, rather than having several syntaxes. I believe he was right to insist on this approach, although his own theories took a winding road in the matter. Remember the kernel sentences versus derived ones, the optional versus obligatory transformations, cyclicity and rule ordering, move-everything versus move and merge, deep and surface structures versus interfaces. The recent convergence of radical lexicalism and minimalism suggests that we have now less degrees of freedom to hypothesize about possible categories, therefore about possible grammars, and we can import each other's results.

The amount of semantics we expect to squeeze out from the syntactic categories is crucial in this debate. If we keep our radar too narrow, as in Chomskyan transformationalism, it seems that we need to make an array of auxiliary assumptions. If we open wide, we would have no choice but do syntax with semantic types, and not even Montague went as far as that. The middle ground seems to be the rule-to-rule-hypothesis of Bach, which was implicit in Chomsky's early writings, which, once radically lexicalized, puts a natural limit to what kind of semantic types can be put in correspondence

with the syntactic ones. It seems that adjacency can serve both ends without extra assumptions.

It is also worth noting that, if the next simplification in transformationalism is the elimination of *move*, as some practitioners of the theory have already proposed (Epstein et al. 1998), then what we will get is essentially some version of categorial grammar, modulo morphology. (Distributed Morphology seems to fill that gap nicely, although its computational properties are understudied at the moment. Autosegmental morphology of McCarthy 1981 is better-known in this respect; see e.g. Bird and Ellison 1994, Kaplan and Kay 1994.) Epstein et al. indeed acknowledge that on the semantic side the states of affairs would look very much like a Montagovian categorial grammar (Epstein et al. 1998: 13), but with some effort to bring in the syntactic instructions about compositional semantics as a residue of derivations, namely the cyclic delivery of partial results. The point of CCG is that semantics is available at any time.

Four independent developments, namely Chomsky's formalized notion of grammar, Lambek's inauguration of radical lexicalism, Schönfinkel and Curry's conception of semantics in order, and Steedman and Szabolcsi's syntacticization of the same transformed Chomsky's 'rule of grammar' to 'category of a word', and 'knowledge of language in grammar' to 'knowledge of words'.

Radical lexicalism as first demonstrated in the 1960 conference by Lambek grew out of the unification of the grammatical description, and its predictive powers for possible linguistic categories far outweighed the simplicity and elegance arguments of the multi-level syntactic approaches with "purer" strata. It is largely a theoretical debate which is not supported computationally.

In turn, computationalism as manifested in the combinatory knowledge of words puts some flesh in Wittgenstein's theory of meaning-is-use, by reflecting a personal history of word usage, both personally and per word, its potential misunderstandings, but no misrepresentation of it. Fallible knowledge is genuine knowledge explicitly represented in a category. The result that it must incorporate some detailed statistical knowledge in tandem with combinatory knowledge should not be surprising to anyone who has followed the research in language acquisition, machine learning and computational linguistics.

We can now go back and study Quine's appraisal of Schönfinkel's work. I repeat Quine's commentary cited in the introduction:

It was letting functions admit functions generally as arguments that Schönfinkel was able to transcend the bounds of the algebra of classes and relations and so to account completely for quantifiers and their variables, as could not be done within that algebra. The same expedient carried him, we see, far beyond the bounds of quantification theory in turn: all set theory was his province. His *C,S,U* and application are a marvel of compact power. But a consequence is that the analysis of the variable, so important a result of Schönfinkel's construction, remains all bound up with the perplexities of set theory. Quine (1967: 357)

His own solution to variable elimination, Quine (1966), needed a meta-theory to avoid the problems he had pointed out, whereas Schönfinkel's theory was an object-level theory, which led to direct syntacticizability without levels or strata. His understandable concerns for set theory are not imported into this syntacticization, because this is combinatory syntax, not set-theoretic. Semantic objects are not sets but predicate-argument structures embodying semantic dependencies, which are structural domains in need of a primitive for construction.

By direct import from the elimination of variables at object language, constituents are built by syntacticization of the same primitive. This might help us see the sister theories of CCG such as Construction Grammar (Goldberg 1995, Croft 2001) and Dependency Grammar (Hays 1964, Hudson 1984, Mel'čuk 1988, Kuhlmann and Nivre 2006) as wanting an explanation why we have the constructions we observe in languages, and why we see only certain kinds of dependencies and constituencies. I have exemplified quite a number of the last kind, ranging from traditional constituents such as VP, NP etc., but also the unorthodox strings that seem to have immediately interpretable subpieces thanks to combinators, such as *I say three mathematicians in ten and you claim four philosophers in five prefer corduroy*, or *I can, and perhaps you will, try to sing 'Flaming.'*

The combinatory process has its limits because it cannot make a compositionally uninterpretable fragment a constituent. It cannot call a fragment a constituent and not immediately deliver a compositional meaning for it. *I can you sing* is a word salad although some parts of it are not, and *I can you* is parasitically interpretable in a gapping environment such as Barrett's *You can't see me, but I can you*.

Whether that makes it a constituent is hard to tell from a combinatory perspective, because the point of combinatory semantics arising from order alone is that each constituent has the stuff to deliver whatever (partial) mean-

ing is available. No such doubts arise about the bracketed substring in (*three mathematicians in*) *ten*; it is a nonconstituent.

There are impossible words too, such as those with **Y** semantics, and suspect words, for example with **K** semantics. Some dependencies are more unlikely than suspect, given the other assumptions of a lexicalized grammar. For example, it is hard to conceive how *John expects that Barry* could mean ‘John expects Barry to expect’. For it to mean that we need **S** semantics where *expect*’-like verbs can be the targets of parasitic extraction, in pseudo-English something like *expect_i from me that I imagined to _i without wishing to _i*. Noun extraction is common, but verb extraction, especially of this kind, is unattested. The theory aspires to be explanatory by being as specific as it can about impossible constituents, and showing explicitly how the possible ones can be constructed. Unlikely ones are a conspiracy of the types in a radically lexicalized grammar. In a way, the grammar as a whole symbolizes making sense of the world of words in their possible combination.

CCG’s neo-Humean answer to the natural limit on constituency and dependency is that all syntactic behavior arises from the self-limiting nature of codetermination of syntax and semantics in a radically lexicalized grammar which faces limited combinatory possibilities. That is all adjacency can offer with less than a handful of noninter-definable dependency encoders and a fully lexicalized grammar.¹⁰⁵ Furthermore, we get the immediate assembly of dependency structures for free by the process of syntacticization, and that should be a good thing.

The emerging **BSO** family epitomizes composition because it is of the form $\lambda x.f \cdot \cdot (g \cdot x \cdot \cdot)$ in binary. The members of the family represent action orientation (the predicates are known), and object opacity (the argument is abstracted over). They are also known as sequencers. The other family, **T**, represents action opacity and object orientation because it is of the form $\lambda P.Pobject'$. It is not a sequencer, but it is a facilitator of sequencing, as the monadic perspective showed. Steedman (2002) relates the first family to action planning, and the second to affordances.

Taken together with the other ingredients of human cognition, most importantly, awareness of other minds and their affordances, they provide a simple ground for semantic recursion and its syntactic reflex without entangling ourselves in the debate about the necessity of syntactic recursion (recall the lack of the **YKI** family among the potential candidates for syntacticized dependencies, which means that, at least in theory, syntactic recursion is not necessary to capture semantic recursion). Therefore, it seems possible that language and

other cognitive activity in primates can be related evolutionarily if seriation is the key.

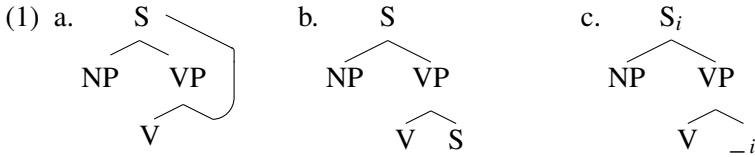
I will close the book by projecting back in time about adjacency. A speculation-wary reader might consider this point to be the end of the book. I will be drawing on some proposals and add a bit of speculation of my own about whether this alternative foundation for grammar—order and its semantics giving us limited constituency and dependency in syntax, has something to add to the studies on language evolution.

Perhaps, but with a caveat, and with some hesitation. First we must remember that Darwin had called his book *The origin of species*, not *The origin of life*. The diversification and evolution of languages once we have acquired the hereditary capacity for language with big L appears to be a different matter than how this seemingly unique capacity came about in the first place; see for example Knight, Studdert-Kennedy and Hurford (2000) for extensive discussion, and the ensuing debate. I will concentrate on the emergence of language with big L.

Take Chomsky's views on the topic, which suggest no intermediate forms of language, Bickerton's (1990) saltational view, Jackendoff and Pinker's (2005) Baldwinian adaptationist view, and Deacon's (1997) Baldwinian view without a universal grammar. The first three arise from the syntactic structure-dependence of syntax, and Deacon's view seems congenial to the emergence of type-dependence as manifested in all categorial grammars because the word does most of the work.¹⁰⁶ Recall that knowledge of words is not a simple competence of lexical look-up in the present discussion; it is combinatory knowledge, that is, a piece of syntax.

Chomsky's view is not surprising because the phrase structure tree with possibly empty elements in it seems to be such a unique source (not even the transformationalist lexicon is constructed from the same source), we can hardly expect to see some precursors or progeny in other cognitive activities.

Recall also that recursion is everybody's assumption in semantics, and syntactic recursion is something we can live without. It is unhelpful to take syntactic recursion as an empirical fact and build a theory of language on it, including its evolution (see Hauser, Chomsky and Fitch 2002). Genuine syntactic recursion is depicted in (1a) alongside semantic recursion (1b) to show the difference. (1a) is a direct syntacticization of **Y** semantics whereas (1b) is semantic recursion as a tree. Note also that (1c) is not the same as (1a); one is an anaphoric dependency and the other is a recursive dependency. It seems safe to say that no language has demonstrated a dependency of type (1a).



Bickerton's (1990, 1996) protolanguage might appear to be similar to adjacency, perhaps to the applicative fragment without combinatory dependencies, but that fragment also gives us context-free dependencies as Bar-Hillel, Gaifman and Shamir (1960) proved. Maybe that is what it was, maybe not. It seems to go over and above what Bickerton intended as protolanguage, because we have reasons to believe that context-free dependencies go a very long way in capturing most of the dependencies we see in today's languages; GPSG was one bold attempt at this task (see Gazdar et al. 1985). The argument-taking fragment sketched in the beginning of the book does not seem to be the niche for protolanguage either because it arises from the same base as combinators, which makes it unlikely that the emergence of language as a combinatory faculty is saltational as Bickerton suggested.¹⁰⁷

Although Chomsky, Bickerton and Pinker differ in many ways about the origins of language, they share the same assumption that universal grammar, for them being a language-specific set of instructions about syntax, grows into an adult-state grammar from an initial state. The knowledge in the universal grammar must include—as of 2009: the syntactic principles, merge, move, check, select, numerate, empty category governance, functional categories and their management, syntactic structure-dependence, and several parameters, either abstract or cognitively realized—the latter variety is endorsed explicitly in popular writing (Baker 2001, Yang 2006). We should assume that it comes with some allotment for bilingualism and trilingualism, along with some precautions for potential conflicts among parameter values or in their order of valuation—recall that there are arguments for a universal order of parameter setting by Baker.

The computationalist alternative to parameter setting is the exponential decay of probabilities as experience is accumulated, not over a long period, but within the confines of a few related experiences, which might give the appearance of a sudden switch setting, as Steedman and Hockenmaier (2007) argued. Some proposed sequencing of parameter valuation, such as the primacy of head-directionality in Baker (2001), has a head start in a radically lexicalized grammar, but not as an on-off switch. It is encoded in every single

linguistic hypothesis about syntactic knowledge of words. (Or we can turn the table around and say that combinatory theory predicts head-directionality to be the primary parameter in a theory of parameters; the lack of clear trends in the setting order of other parameters in Baker's repertoire seems to suggest that they are more about lexical organization, hence about lexical syntactic types, e.g. the ergativity parameter and the serial verb parameter.)

This argument for an alternative view begs the following question: How can we assume every single hypothesis to carry directionality when it is much more convenient to set it for all of them at once? We can calculate a child's potential of making sense of the world if she thinks half the verbs she hears are SVO and the other VSO in a language like English. Insisting on her VSO hypotheses would put her at exponentially increasing risks of gawking at motherese. In the English case, VSO is a clear loser and might show a parameter effect. The survivors happen to have the same head-directionality, without a parameter. For Turkish, this parameter faces problems. OVS, SVO and VSO put together are nearly as common as SOV in child language (Slobin and Bever 1982, Aksu-Koc and Slobin 1985). (Precise numbers are 53%, 37%, and 10%, for SOV+OSV, SVO+OVS, and VSO+VOS, respectively.) The age range for this performance is (2;3-3;8). Ekmekçi (1986) reports that, at (1;10), OV and VO are produced by the child. When children were asked to imitate motherese word order, they were successful 72% (SOV), 60% (OVS), 46% (SVO), 43% (OSV) of the time, at mean age (3;3) (Batman-Ratyosyan and Stromswold 1999). We would expect other parameters to be subsequently effected by this very flexible parameter value, because of the presumed primacy of head-directionality. The problem of charting the precise timing of parameter settings would be replaced in computationalist models by the task of understanding the complex interactions of linguistic hypotheses, assuming a somewhat uniform motherese topics. Directionality will be there from day one.

The computationalist perspective is considered to be a resurgent empiricism of the Humean kind (but not necessarily the Lockean kind—see Machery 2006 for some cogent warnings and extended discussion), in which Hume's associationism is not taken as the inner cause, but as the source of toolboxes in a computational mechanism (of resemblance, contiguity and causation), such as in acquisition and inference. (My personal attempt at these tasks was Bozsahin and Findler 1992, where we relied on, as in the works of others cited there and in the models developed later, the Humean constraints on the hypothesis space.) Combinators too can be naturalized tool-

boxes. Call them spandrels if you like, but crucially, they will be of Dennett's (1995) kind, not Gould and Lewontin's (1979), because they are not necessary mechanisms, just good solutions to a variety of interrelated problems about sequencing.

Combinatory grammar and its radical lexicalization suggest limited invariant combinatorics in lieu of universal grammar. This seems to require a symbolic base (and seriation) which the language must tap on, and perhaps only that. Deacon (1988, 1997) has shown a way how indexical here-and-now knowledge can give rise to internal self-reorganization to lead to symbol systems. Turing (or discrete) representability seems necessary for that, as argued in the book.

Steedman (2002) suggests the involvement of **BT** in planned and coordinated activity in close cousins of ours, crucially without an **LF**, suggesting that **LF** and the syntactic specialization of the combinatory faculty—the syntactic type—might be the source of language. (As I noted earlier, there are disagreements about **LF**.)

If language is a specialization of an earlier combinatory trait (and syntactic types are indeed different than visual, auditory or procedural combinatory categories), then we can expect adjacency to play the key role in this. That of course does not imply that there is evolution *for* grammars, perhaps not even for language. The selection pressure might be for better symbol processing, and more of it.

It seems pointless to expect further exploits of seriation by nature, in the form of syntacticizing the combinators we have so far not seen in natural language syntax. The combinatory path for language, if true, would have had to have been opportunistically selected for a long time, two million years or more.

In this regard, the combinatory view allows us to reassess certain claims about exaptationism and creative use of language, the latter understood to be a product of infinitude. There seems to be no forceful argument to treat them as facts. Exaptationism as an effect (but not as a cause) is already built in to Darwin's theory, as opportunistic selection.

We can put in context the proposals about whether language is a case of exaptation or opportunistic selection. Take for example Yang's (2006) title, *The infinite gift*. As we have seen, infinitude or finitude does not make linguistic theorizing more (or less) exciting, for we will need a theory even if language is vast but finite. Whatever the size and bounds, that theory must be about discrete units—words—if the current reasoning in the book shows

promise. And for discretely representable linguistic knowledge, giving semantics to order, and order alone, to lead to structures in language seems to be a scientifically more conservative start. Likewise, given Darwinian adaptationism and opportunistic selection for combinatory traits, rather than Gould-style exaptationism, it seems that we would earn the language with big L over a long time, rather than take it as an exapted gift.

Appendices

A: Lambda calculus

This appendix briefly reviews lambda calculus. It is not a general or comprehensive introduction to the topic. The material covered relates to the main body of the text and they are used in it frequently.

Lambda terms (equivalently, λ -terms) are well-formed lambda expressions. They are recursively defined as follows.

λ -**words** are constructed from the alphabet

- x, y, z, \dots for variables,
- $1, 2, a', b'$ for invariables (constants),
- λ for abstractor (lambda binding),
- $(,)$ for grouping (parentheses).

λ -**terms** are the set Λ such that

- variables and invariables are in Λ ,
- if $M \in \Lambda$ then $(\lambda x.M) \in \Lambda$ where x is an arbitrary variable,
- if $M, N \in \Lambda$ then $(MN) \in \Lambda$.

By convention, we write multiple lambda bindings with a single dot: $\lambda x.\lambda y.xy$ is written as $\lambda x\lambda y.xy$.

Also by convention, lambda bindings associate to the right, and juxtaposition associates to the left. $\lambda x\lambda y\lambda z.xyz$ is same as $\lambda x(\lambda y(\lambda z((xy)z)))$.

A variable is **free** if it is not in the scope of its lambda binding, **bound** otherwise. For example, x is free in $x + 2$, $\lambda y.x$ and $(\lambda x(a'b'))x$. It is bound in $\lambda x.x$ and in $\lambda x\lambda y.xy$. Within the inner body of the last lambda term, xy , both variables are said to have **free occurrences** because there is no lambda binding in the body.

Lambda conversions are operations that denote equivalences among lambda terms. When used in the direction of eliminating a lambda binding, they are called **reductions**. If a lambda is introduced they are called **abstractions**.

The conversions rely on the property of substitution for bound variables. Eta conversion shows the behavioral equivalence of the typed objects with and without variables. Beta conversion is the main mechanism to establish function application and function abstraction as two sides of the same coin. Alpha conversion shows the equivalence of bound variables under substitution. Together they define equivalence in the function treatment of lambda calculus.

substitution $M[a/x]$ stands for substituting a for free occurrences of x in M .

η -**conversion** $\lambda x.fx =_{\eta} f$, if x is not free in f .

β -conversion $\lambda x.M(a) =_{\beta} M[a/x]$

α -conversion $\lambda x.M =_{\alpha} \lambda y.(M[y/x])$, if scopes of variables in $\lambda x.M$ and $\lambda y.(M[y/x])$ are the same.

equivalence $M = N$ iff $Ma =_{\alpha, \beta, \eta} Na$, for all lambda terms M, N, a .

Read ‘=’ as ‘behaves the same’, not as ‘identical’. From substitution and beta reduction, we get $\lambda x.fx(a) =_{\beta} fx[a/x]$, which is the same as fa , hence the association of beta with function application and abstraction. By equivalence, $\lambda x.fx = f$ too, hence the same behavior when f is supplied with a . The condition on eta conversion ensures that we do not change the behavior of objects; $\lambda x.(\lambda y.yx)x$, in which x is free in $\lambda y.yx$, is not equivalent to $\lambda y.yx$. Similarly, the condition on alpha conversion avoids an accidental capture of the same names, for example $\lambda x.y \neq_{\alpha} \lambda y.y$, and $\lambda x\lambda y.xy \neq_{\alpha} \lambda y\lambda y.yy$.

Normalization refers to the successive application of a conversion until it no longer applies. For example, the beta normalization of $(\lambda x\lambda y.f'yx)(a')(b')$ is two applications of beta reduction giving $f'b'a'$. The eta normalization of $\lambda x\lambda y.f'yx$ is f' . Some lambda terms have no normal forms because the process may not always terminate: $(\lambda x.xx)(\lambda x.xx)$ has no beta normal form.

Normal-order evaluation of a lambda term is the application of beta reduction to the leftmost outermost reducible expression (**redex**) first. In $(\lambda x.x)((\lambda y.y)a')$ there are two redexes, and normal-order chooses to reduce it to $(\lambda y.y)a'$, i.e. the application of the second one, without evaluation, to the leftmost redex. The Church-Rosser theorem establishes the result that two distinct sequences of reductions from the same lambda term will yield the same normal form if there is one. For the example above, it is a' .

B: Combinators

This appendix covers some mathematical aspects of combinators. Much of the book is about turning combinators into linguistic devices for explanation. These aspects are covered in the main body of the text.

Combinators are lambda terms with no free variables. As such they epitomize the compositional behavior of functional objects without a need for variables. By a convention going back to Curry and Feys (1958) they are written as single letters in bold. No extra notation is needed to describe their behavior. The ones considered most basic are defined below. The names were given by Curry and Feys.

B $\stackrel{def}{=} \lambda f \lambda g \lambda x. f(gx)$ (compositor)

S $\stackrel{def}{=} \lambda f \lambda g \lambda x. fx(gx)$ (substitutor)

C $\stackrel{def}{=} \lambda f \lambda g \lambda x. fxg$ (elementary permutator)

T $\stackrel{def}{=} \lambda f \lambda g. gf$ (commutator)

W $\stackrel{def}{=} \lambda f \lambda x. fxx$ (duplicator)

K $\stackrel{def}{=} \lambda f \lambda g. f$ (cancellator)

For example, $\lambda x. a'xx$ is equivalent to $\lambda x. \mathbf{W}a'x$, which is eta-normalizable to $\mathbf{W}a'$.

Combinators established computability about a decade before Turing machines. Their equivalent power can be seen without proof: **K** can delete any sequential material, **S** can expand and compose sequences, **C** can swap their order, which are the basic mechanisms that give the Turing machines their power. In this sense the Turing model is a formal specification of an algorithm in detail, and combinators are its global compositional view.

Normal-order evaluation of combinators evaluates the leftmost outermost combinator first. For example,

$$\mathbf{BSC}fga = \mathbf{S}(\mathbf{C}f)ga = \mathbf{C}fa(ga) = f(ga)a$$

As in the case of lambda calculus, the process may be nonterminating: **WWW** evaluates to itself indefinitely.

For the sake of completeness, I list the well-known combinators in Table 7. The names in the table are from Smullyan's (1985) tale of combinators as singing birds. They are in common use as well.

As Curry, Feys and Smullyan note, there are many equivalences between the combinators. This aspect opens way to linguistic theorizing about which must be included in the grammar or in the lexicon, therefore they belong to the main body of the book.

Table 7. Some well-known combinators

I	$Ix = x$	Identity bird
Y	$Yx = y = xy$ for some y depending on x	Sage bird
U	$Uxy = y(xxy)$	Turing bird ¹⁰⁸
K	$Kxy = x$	Kestrel
T	$Txy = yx$	Thrush
W	$Wfx = fxx$	Warbler
B	$Bxyz = x(yz)$	Bluebird
C	$Cxyz = xzy$	Cardinal
S	$Sxyz = xz(yz)$	Starling
Φ	$Φxyzw = x(yw)(zw)$	
Ψ	$Ψxyzw = x(yz)(yw)$	
J	$Jxyzw = xy(xwz)$	Jay

The **power** of a combinator is a generalization of its behavior. For example, $B^n f$ composes f with n -argument functions, whereas **B** composes two one-argument functions. It is defined as follows:

$$X^0 = \mathbf{I},$$

$$X^1 = X,$$

$$X^n = \mathbf{B}X X^{n-1} \text{ for } n > 1, \text{ for a combinatory object } X.$$

Therefore, $B^2 fgab = \mathbf{B}Bfgab = \mathbf{B}(\mathbf{B}f)gab = \mathbf{B}f(ga)b = f(gab)$. Powers are not distinct combinators, and they serve a crucial role in generalizing the linguistic notion of arity.

A **supercombinator** is a combinator in normal form in which all its argument-taking lambdas (its lambda bindings) can be grouped to the left, i.e. its behavior can be made fully transparent looking from the outside. The formal definition is as follows (from Hughes 1984):

Let $\mathcal{S} = \lambda x_1 \cdots \lambda x_n. E$ where E is not a lambda abstraction. \mathcal{S} is a supercombinator of arity n if (a) \mathcal{S} is a combinator, (b) any lambda abstraction in E is a supercombinator, and (c) $n \geq 0$.

In other words, if we can group all bindings before E , and leave no free variables inside E which must be remembered—bound—outside, then we have a supercombinator. Almost all the combinators we have seen so far are supercombinators, but not all combinators are supercombinators. The function $\lambda y.y(\lambda x.yx)$ is not a supercombinator because y occurs free in the inner lambda term. Supercombinators will directly relate to the argument-taking behavior of the linguistic notion of ‘head of a construction’.

Fixpoint combinators stand out of supercombinators because they allow us to capture recursion without use of names or variables. One such combinator is **Y**. Its

definition is given below. Note that **Y** is not a supercombinator. It finds the fixpoint of any function h , as shown.

$$\mathbf{Y} \stackrel{\text{def}}{=} \lambda h. (\lambda x. h(x x)) (\lambda x. h(x x))$$

$$\mathbf{Y} h = h (\mathbf{Y} h)$$

It is truly remarkable that with the use of **Y**, recursion can be achieved without names. I borrow from a classic in the field of programming, Peyton Jones (1987: §2.4), to tell the story.¹⁰⁹

Consider the following definition of the factorial function, where recursion is explicit due to naming (which is something we cannot do in lambda calculus).

$$\text{FAC} = \lambda n. \text{IF} (= n 0) 1 (\times n (\text{FAC} (- n 1)))$$

This recursive definition can be turned into self-application without recursion as below, because of beta conversion. Note that **H** is not recursive.

$$\text{Let } \mathbf{H} = \lambda f \lambda n. \text{IF} (= n 0) 1 (\times n (f (- n 1)))$$

$$\text{Then } \text{FAC} = \mathbf{H} \text{ FAC}$$

$$\text{because } \text{FAC } n =_{\beta} \mathbf{H} \text{ FAC } n \text{ for any natural number } n \geq 0$$

The point of course is to be able to recurse without names on any function, not just the factorial. This is where the combinator **Y** can help. The factorial can be defined without recursion or names. The steps below are borrowed from Peyton Jones (1987: 27). They show that it does the equivalent of the recursive factorial.

$\text{FAC} = \mathbf{Y} \mathbf{H}$, where **H** is as defined above.

$$\text{FAC } 1 =$$

$$\mathbf{Y} \mathbf{H} 1 =$$

$$\mathbf{H} (\mathbf{Y} \mathbf{H}) 1 =$$

$$\lambda f \lambda n. \text{IF} (= n 0) 1 (\times n (f (- n 1))) (\mathbf{Y} \mathbf{H}) 1 =$$

$$\lambda n. \text{IF} (= n 0) 1 (\times n (\mathbf{Y} \mathbf{H} (- n 1))) 1 =$$

$$\text{IF} (= 1 0) 1 (\times 1 (\mathbf{Y} \mathbf{H} (- 1 1))) =$$

$$\times 1 (\mathbf{Y} \mathbf{H} 0) =$$

$$\times 1 (\mathbf{H} (\mathbf{Y} \mathbf{H}) 0) =$$

$$\times 1 ((\lambda f \lambda n. \text{IF} (= n 0) 1 (\times n (f (- n 1)))) (\mathbf{Y} \mathbf{H}) 0) =$$

$$\times 1 ((\lambda n. \text{IF} (= n 0) 1 (\times n (\mathbf{Y} \mathbf{H} (- n 1)))) 0) =$$

$$\times 1 (\text{IF} (= 0 0) 1 (\times 0 (\mathbf{Y} \mathbf{H} (- 0 1)))) =$$

$$\times 1 1 =$$

$$1$$

The problematic property of **Y** is that it cannot be reduced to a form which cannot be reduced any further, thus the only way to stop recursion by **Y** is to reach nonrecursive (base) cases, such as reaching the ‘ $\times 1 1$ ’ step above.¹¹⁰ Below is **YK**’s infinite expansion. The base cases are an infinite supply of semantic objects following **YK**.

$$\mathbf{YK} = \mathbf{K}(\mathbf{YK}) = \mathbf{K}(\mathbf{K}(\mathbf{YK})) = \mathbf{K}(\mathbf{K}(\mathbf{K}(\mathbf{YK}))) = \dots$$

In the book I follow the convention of writing a syntacticized combinator with its arity as a prefixed subscript. The subscript will be omitted when the arity is same as its combinatory definition, for example 2 for **T** and **K**, 3 for **B**, **S** etc. Curry and Feys (1958) use the notation $(X)_n$ for the same purpose where n is the arity, but the use of

parentheses for that purpose is somewhat unfortunate because they do so much work on the right-hand side of the definitions. Other options such as X_n , X^n , $X_{(n)}$, $X_{[n]}$ are used for other purposes by Curry and Feys (1958). I note the convention for easy reference:

For a combinatory object X ,
 its arity k in a particular use is denoted as ${}_kX$. (arity-in-use)
 Arity is omitted when it is the same as in X 's definition.
 For example, ${}_2\mathbf{T}$ is same as \mathbf{T} . ${}_2\mathbf{B}$ is binary use of ternary \mathbf{B} .

C: Variable elimination

There is nothing any theory can do if a variable is free to vary. The process of variable elimination therefore relates to bound variables. It can be done in various way, as Frege, Schönfinkel, Geach and Quine have shown. This appendix is concerned only with the possibility of variable elimination. (The manner in which it is done bears on linguistic theory, and is dealt with in the main body.)

First we note that if all bound variables of a function symbolize the applicative behavior of the function, i.e. if they are used in the order they are lambda-bound, and only once, then eta conversion can do all the work, as follows.

$$\begin{aligned} \lambda x_1 \cdots \lambda x_{n-1} \lambda x_n. f x_1 \cdots x_{n-1} x_n & \text{ equals, by associativity, to} \\ \lambda x_1 \cdots \lambda x_{n-1} \lambda x_n ((f x_1 \cdots x_{n-1}) x_n) & =_{\eta} \\ \lambda x_1 \cdots \lambda x_{n-1}. f x_1 \cdots x_{n-1} & =_{\eta} \\ \vdots & \\ \lambda x_1. f x_1 & =_{\eta} \\ f & \end{aligned}$$

Therefore eta conversion is equivalent to saying that all semantic invariants are inherently typed. Once we know that f is say a three-argument function with applicative behavior, then writing f^3 or just f is sufficient.

The rest of the dependencies, for example $\lambda x.fxx$ or $\lambda x.f(gx)$, are not eta-normalizable without the help of combinators. For example, the first one of these is eta-normalized as $\lambda x.fxx = \lambda x.\mathbf{W}fx =_{\eta} \mathbf{W}f$ and the second $\mathbf{B}fg$. Schönfinkel's work showed that two suffice for this task, because \mathbf{S} can be seen as a mechanism of pushing the lambda bindings inside, which will eventually reach a base case such as $\lambda x.x$, $\lambda x.y$ or $\lambda x.a'$, which are lambda terms with the simplest body of functions. These properties follow from the following equivalences.

$$(\lambda x.MN)a =_{\beta} \mathbf{S}(\lambda x.M)(\lambda x.N)a$$

hence $(\lambda x.MN) = \mathbf{S}(\lambda x.M)(\lambda x.N)$ from equivalence in lambda calculus.

The elimination is completed by the following equivalences:

$$\lambda x.y = \mathbf{K}y \quad \lambda x.a' = \mathbf{K}a'$$

$$\lambda x.x = \mathbf{I}$$

The equivalences are applicable to any lambda-definable (hence Turing computable) object. For example, $\lambda x.MNP$ is equivalent to $\lambda x.(MN)P$ because of left-associativity, thus any number of lambda terms can be handled by \mathbf{S} . In case of multiple abstractions such as $\lambda x\lambda y.love'xy$, we have to apply \mathbf{S} -pushing to the innermost lambda first.

Knowing that $\mathbf{I} = \mathbf{SKK}$, we can eliminate all bound variables and write everything in terms of \mathbf{S} , \mathbf{K} and the invariables. For example, everything except hit' and $john'$ can be eliminated from the following formula.

$$\lambda x.hit'xjohn' =$$

$$\begin{aligned}
 & \mathbf{S}(\lambda x.hit'x)(\lambda x.john') = \\
 & \mathbf{S}(\mathbf{S}(\lambda x.hit')(\lambda x.x))(\mathbf{K}john') = \\
 & \mathbf{S}(\mathbf{S}(\mathbf{K}hit')\mathbf{I})(\mathbf{K}john') = \\
 & \mathbf{S}(\mathbf{S}(\mathbf{K}hit')(\mathbf{S}\mathbf{K}\mathbf{K}))(\mathbf{K}john')
 \end{aligned}$$

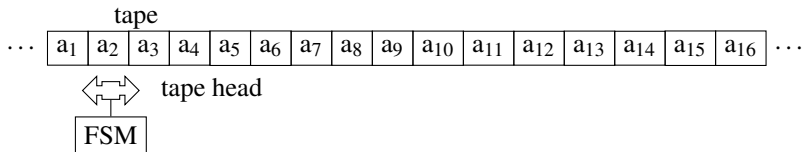
This is a dangerous practice because of **K**'s powers of deletion. The reader can verify that the following formula works endlessly to reproduce itself, due to having both **S** and **K**. Some steps are shown.

$$\begin{aligned}
 & \mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I})(\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I}))(\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I})) = \\
 & \mathbf{S}(\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I}))(\mathbf{K}\mathbf{I}(\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I}))) (\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I})) = \\
 & \mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I})(\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I}))(\mathbf{K}\mathbf{I}(\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I}))) (\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I})) = \\
 & \vdots \\
 & \mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I})(\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I}))(\mathbf{I}(\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I}))) = \\
 & \mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I})(\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I}))(\mathbf{S}\mathbf{S}(\mathbf{K}\mathbf{I})) \\
 & \vdots
 \end{aligned}$$

D: Theory of computing

The theory of computing features quite often in the book because it has empirical and theoretical consequences for combinatorial linguistics. In the first aspect, the children seem to be facing a computationally tractable problem in language acquisition and stagewise development. Granted that there have been some warnings about using the algorithmic complexity theory at face value for this task (e.g. Berwick and Weinberg 1982), the narrower claim of the theory covered in the book is that a performance grammar is competence grammar because it delivers the immediate assembly of all constituents and their meanings, partial or full. Theoretically, another aspect of algorithmic computation seems very relevant to natural language: discrete representability, without which complexity theory is meaningless. Because Turing (1936) was the first to give us a view of functions unheard of before, as a step-by-step computing over a representation, I will refer to it as Turing representability. The appendix covers these aspects very briefly from a mathematical perspective.

A **Turing Machine** (TM) is a finite-state abstract machine with an unlimited supply of sequential memory (usually called “tape”) to which it can write, rewrite and scan one cell at a time:



A tape cell may contain a symbol a_i or it may be blank. The FSM is the finite-state machine component. Before computation starts, a TM is in the start state of the FSM with the tape head pointing to the beginning of the input, if any, and the remaining cells are assumed to be blank. It stops when it reaches a “halt state” in the FSM (there are many alternative definitions; this one suffices for our purpose of computing a function).

It has no notion of physical timing; its measure of a problem size is a combination of the number of states and the number of steps it takes to compute a function. We can assume that every basic step (read, write, rewrite or move left or right on the current tape head, and/or change the current state in the FSM) takes a constant time, but that is in theory unnecessary; it might as well happen simultaneously. What matters is that taking the next step requires the notion of “next”, and that is either one cell, one symbol or one state, so that once we take the step we will have moved one step more than the earlier status in some regard above. These are the bases of complexity measures in the theory of algorithms.

A **configuration** of a TM is a collection of its current state, the current pointer to a cell in the sequential memory, and its memory content. Trailing blank cells are

not considered part of the content. Memory content can of course be indefinitely stretched, which we can capture as a regular expression.

A Deterministic Turing Machine (DTM) is a TM in which every configuration is uniquely determined by the previous one. This is Turing's capture of the notion of function in a step-by-step manner. If there is more than one way to take the steps of the function, a DTM can simulate these choices by making use of another tape to keep track of its moves while checking whether they all agree on the result, which we can keep on yet another tape. This is Turing's capture of the notion of relation, which is a function over powersets of inputs and outputs. We know that multi-tape Turing machines and other variations such as multiple tape heads, nondeterminism, random-access memory rather than the tape do not give us more things to compute than a standard TM (see Hopcroft and Ullman 1979, Lewis and Papadimitriou 1998 for these results).

A TM is said to be **nondeterministic (NDTM)** if it can make a "guess" of the solution and check (as a DTM) whether it is indeed a solution. We can take the guessing stage to be equivalent to putting on another tape the precise sequence of steps to follow. In this regard we do not get a new class of computation but a new class of how to do computing, i.e. a complexity measure.

An algorithm is a DTM that always **decides**, i.e. if it can stop for any input to make a decision. A nondeterministic algorithm does the same with a NDTM. A **procedure** (or heuristic) is a DTM or NDTM that **semi-decides**, i.e. if they can stop on some input to make a decision.

Undecidable problems are "functions" for which there is no algorithm (deterministic or nondeterministic). The Halting Problem of the Turing machine in which a TM takes as input another TM and tries to decide whether it stops on all its inputs, is one such problem. The problem is at least formulable, but it is not solvable. Some problems are expressible but not formulable, for example: "what is the next number after π ?"

In the book a problem will be called **Turing-representable** if it can be written as a TM (but not necessarily solved by it). For example, the Halting Problem is Turing-representable as below (from Lewis and Papadimitriou 1998; $halts(P,X)$ means P halts on input X). It is the *diagonal(diagonal)* program. The π question is not Turing-representable.

diagonal(X):

a: if halts(X,X) then goto a else halt.

Turing-representability ties in with another line of development that gave us the understanding of limits of computability today: the recursion theory. **Primitive recursive functions** are those which can be defined by identity, succession, composition and recursion. The successor function $succ(n) = n + 1$ is crucial in this definition, which gives us the link to Turing-representability by providing a notion of "next".

A computationally tractable problem is one for which there is an algorithm that works on a polynomial function of the size of the problem for a DTM (i.e. its number

of states, the number of steps it must take and the space it must use, as a function of the Turing-representable input). The complexity class \mathcal{P} symbolizes such problems. Computing scientists sometimes use the term “polynomial time function” to talk about these problems, and care must be taken not to misunderstand the word *time*. It does not measure the physical time or space but abstract time and abstract space, which are the abstract measures of problem “size” from Turing representations. (In this sense computation as we know today cannot be a natural law as Chomsky once suggested.)

A **computationally intractable** problem is one for which there is a NDTM that can guess a solution and check its validity in polynomial time. This is a very important class of complexity, called \mathcal{NP} , for “nondeterministically polynomial”. Intractable problems, then, have an exponential algorithmic solution, all of which can be checked in polynomial time individually.

The order of a function limits its behavior on the abstract size of the problem “from above.” The order of f is g , written $f = O(g)$ by convention, if for some positive constants c and n_0 , $f(n) \leq cg(n)$ for all $n > n_0$. If f is n^2 it is $O(n^3)$ and also $O(n^4)$ etc. It is $O(n^2)$ too, but n^4 is not $O(n^2)$ or $O(n^3)$. This notation allows us to equate \mathcal{P} problems with $O(n^k)$ order, for some constant k , and the \mathcal{NP} class with $O(n^n)$, where n is the problem size in the Turing sense.

Many interesting problems are \mathcal{NP} , e.g. finding the possibility of the truth of a set of disjunctive logical formulae such as $A_1 \vee A_2 \vee A_3$ and $\neg A_1 \vee A_2 \vee A_3$. If we are given the truth conditions of A_i , we can check in polynomial time whether the set is satisfiable (i.e. true in all its clauses). If not, we must check every truth assignment, which is exponential on the size of the set, therefore computationally intractable.

The fact that we do know this even if generating the entire solution space may wear us down relates the notion of Turing-representability, algorithms, competence and performance at the abstract level rather than concrete. This is the significance of the theory of computing for linguistics. It is an intensional body of knowledge.

In this regard a computational look at language cannot be understood just by looking at problem complexity, timing or space through the classes \mathcal{P} and \mathcal{NP} . The approach and these complexity classes are intrinsically tied to abstract and discrete representability, which translate to scaling up of the knowledge of competence and identifying similarly characterizable problems of cognition.

We may compare a computational solution with a noncomputational one to see the nature of the argument. Consider sorting n quantities, say a sheaf of spaghetti rods to be sorted by length. A noncomputational solution in the sense of avoiding a Turing representation might be to conceive them as physical quantities such as weight, length and solidity. Sorting can be done with a variant of Dewdney’s (1984) method, which is itself algorithmic and linear. We take the sheaf of spaghetti cut to different lengths, where length represents itself, i.e. an approximation of the quantity along which we sort. We bang the sheaf on the table and pick the ones that stick out progressively. This is in principle instantaneous if we leave the sorted spaghettis

in place rather than separate them. In contrast, a computational solution would be to map the quantities to some representation, say numbers, and solve the problem as a case of sorting anything that has a discrete representation, which is $O(n \log n)$. In the first case we can claim to have understood gravity, solidity and eye measurement. In the second case we understand the nature of the problem. The first solution would not scale up even if we assume to have devised a representation of weights through spaghetti and tables because it is not translatable, it will not work in outer space, or for gases. We might search for a mapping of any problem so that gravity can solve it by natural laws, but in doing so we would be turning gravity into a computer, crucially one that works over a representation, which is the mapping itself. We can compare this approach to the original analog algorithm of Dewdney for spaghetti sort, which is indeed an algorithm therefore a computational solution because although it makes use of gravity to sort the spaghetti rods, it iterates on the broken spaghetties for sorting, hence its complexity measure is not the physical time associated with gravity but the number of steps. (I am grateful to Mark Steedman for suggesting a look at Dewdney.)

E: Radical lexicalization and syntactic types

Radical lexicalization refers to the process of rewriting all the rules in a phrase-structure grammar which do not make reference to a lexical item on the righthand side, as rules for the lexical items. These rules collectively become the lexical item's combinatory category. Two kinds of phrase-structure rules, context-free rules and linear-indexed rules, can always be given such a treatment.

A **linear-indexed grammar** (LIG) is a context-free grammar equipped with a stack such that the lefthand side of rules can push, pop or pass the stack to the righthand side, and only to one symbol on the right (hence the term "linear"). Such grammars can generate strictly noncontext-free languages. For example, the grammar below generates $\{a^n b^n c^n \mid n \geq 0\}$ ('..' denotes the remainder of the stack '[']').

$$\begin{aligned} S[..\] &\rightarrow aS[..b]c \\ S[..\] &\rightarrow A[..\] \\ A[..b] &\rightarrow A[..\]b \\ A[\] &\rightarrow \varepsilon \end{aligned}$$

This appendix shows the radical lexicalization of a context-free grammar. Linear-indexed grammars are related to CCG hence covered in the main text.

Let us consider the following fragment of a context-free phrase-structure grammar to clarify the process. Exclusive terminals in the second column stand for the lexicon, and the grammar rules on the left refer to substantive categories S, NP, VP, V etc.

S	→	NP VP	Det	→	every
NP	→	Name	N	→	chemist
NP	→	Det N	Name	→	Kafka
VP	→	V _{iv}	V _{iv}	→	arrived
VP	→	V _{tv} NP	V _{tv}	→	adored

First, the information about arity is redundantly specified in this grammar. The rule $VP \rightarrow V_{tv} NP$ specifies that the verb is transitive because there must be an NP following the verb, and the lexical entry by the preterminal V_{tv} duplicates that information. We can take the rule to mean that a transitive verb, once it takes an NP to the right, yields a VP. That is, $V_{tv} = VP/NP$ in present terms. We could also write $NP = VP \setminus V_{tv} = (S \setminus NP) \setminus ((S \setminus NP) / NP)$, because from the S rule we can write $VP = S \setminus NP$.

Similarly, $V_{iv} = VP$. Because the NP rules have lexical anchors in this grammar (name and determiner), we can follow the same strategy and arrive at $Det = NP/N$ and $Name = NP$. We could also write $N = NP \setminus Det$ if we wished. The S rule has no lexical anchor, thus we must write it as both $NP = S/VP$ and $VP = S \setminus NP$. We have arrived at the following equivalences:

$$\begin{array}{lll}
 V_{tv} = VP/NP & V_{iv} = VP & NP = VP \setminus V_{tv} \\
 NP = S/VP & VP = S \setminus NP & Det = NP/N \\
 Name = NP & N = NP \setminus Det &
 \end{array}$$

$$\begin{array}{ll}
 \text{Hence } V_{tv} = (S \setminus NP)/NP & V_{iv} = S \setminus NP \\
 NP = (S \setminus NP) \setminus ((S \setminus NP)/NP) & \\
 NP = S/(S \setminus NP) &
 \end{array}$$

We can eliminate the phrase-structure rules in the left column of the phrase-structure grammar above, and write only the lexical items with their new categories, to capture the same fragment of English surface syntax:

$$\begin{array}{ll}
 \textit{every} & := \text{Det} = NP/N = (S/(S \setminus NP))/N \\
 \textit{chemist} & := N = NP \setminus \text{Det} = NP \setminus (NP/N) \\
 \textit{Kafka} & := \text{Name} = NP = S/VP = S/(S \setminus NP) \text{ and} \\
 & \quad (S \setminus NP) \setminus ((S \setminus NP)/NP) \\
 \textit{arrived} & := VP = S \setminus NP \\
 \textit{adored} & := VP/NP = (S \setminus NP)/NP
 \end{array}$$

What we cannot eliminate, of course, is the right column because that would change the empirical coverage of the grammar.

Any context-free phrase-structure grammar and linear-indexed grammar can be reduced to its lexicon if we are willing to translate the distributional categories such as N, V, A, P to combinatory categories as above. We can do this because any rule in these formalisms has one symbol on the left-hand side, with or without a stack, therefore a functional reading of the rule from right to left is always possible. (LIGs do not distribute a stack on the right, therefore the compositional reading of a LIG rule is straightforward too.) Notice also that the redundancy of V_{tv} specification has disappeared in the course of the translation.

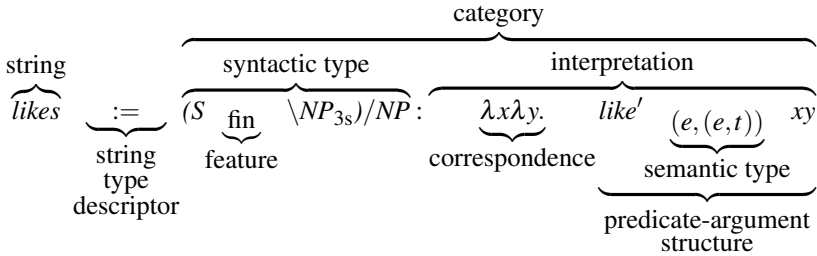
One can argue that the elimination of unwanted ambiguity leads to another ambiguity, viz. $NP = (S \setminus NP) \setminus ((S \setminus NP)/NP)$ and $NP = S/(S \setminus NP)$. We shall see in the text that the newly introduced ambiguity is not spurious; it relates to case marking.

A combinatory syntactic type can be thought of as a collection of the applicative translation of all phrase-structure rules as above, plus their combinatory derivatives. For example, from $S/(S \setminus NP)$ and $(S \setminus NP)/NP$ in this order we also get S/NP because of composition. They can be thought of as the possible landscape of all types derived from the lexical items as a closure of the lexicon on combinators. A linguistic theory will select a subset in some principled way.

An example type is shown below.

$$\textit{likes} := (S_{\text{fin}} \setminus NP_{3s})/NP : \lambda x \lambda y. \textit{like}'xy$$

The breakdown of its constituents is in the next page. Additionally, I use a common index as a simple way to share the common features among syntactic types, for example $\textit{word} := S_i/(S_i \setminus NP_{3s \in i})$. The i here is a shared set of features among which there is the third-person singular emanating from the NP . To avoid notational clutter, this convention is suppressed when it is not critical to the discussion. Feature abbreviations are also quite common in the book, to write NP_{3s} to mean $NP_{\text{AGR}=3s}$.



When no confusion arises, I will use the term **category** for the combinatory syntactic type.

A consequence of radical lexicalization is that one end of the rules for the lexical items is the syntactic type, and, since there is no other loci if lexicalization is strictly followed, then the other end has to be a predicate-argument structure, which bears the semantic types. I cover the consequences of this result in the main text.

A **semantic type** is a narrowing of a predicate-argument object in possible values. The type e is for things (Montague's *entity*), t is for propositions, and (e, t) is for predicates and properties, that is, for functions from things to propositions. For example the transitive verb *like*, with the semantics $\lambda x \lambda y. \text{like}' xy$, has the semantic type $(e, (e, t))$. The eta-normalized version like' is assumed to carry this type along. Thus, like' is not of type t , which its bare form might suggest. In that sense, every semantic object has a type.

F: Dependency structures

Dependency structures may be specified over words in a string in some theories and over predicate-argument structures in others. This topic belongs to an appendix because it can be done without combinators. With combinators, it is defined over a predicate-argument structure, and this narrow view is explained here.

A **dependency structure** is a relation between two semantic objects. For our purposes it can be defined as follows.

A function depends on its arguments. (dependency)

Juxtaposition xy means ‘ x depends on y ’ (juxtaposition)

It arises from a functional interpretation of the concept. (I use a distinction, more commonly made in computer science and computational linguistics, between functions, predicates and algorithms. The term *function* refers to opaque properties, such as arity, dependence and output, whereas the term *predicate* refers to transparent properties such as the event class, argument structure and their obliqueness, although, formally speaking, they are both functions or relations. *Algorithms* are functions that do something. I will use this term when we are interested in the task of the function rather than its dependencies or structures.)

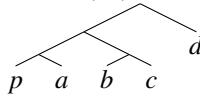
A **predicate-argument dependency structure**, abbreviated to **PADS** in the book, is a predicate-argument structure of dependencies where the leftmost element is a predicate. For example, $john'mary'$ is a dependency structure but not a PADS, and $sleep'john'$ is a dependency structure and a PADS. As we shall see throughout the book, PADS is different from the logician’s logical form, from the transformational linguist’s logical form, from the dependency structures between words of a string, and from model-theoretic objects. It is lexically determined and projected. It depends on the syntactic type in crucial ways, codetermines it in crucial ways, and it is indeed a nonassociative structure: $(hurt'love')mary'$ is different than $hurt'(love'mary')$. In the first case, $mary'$ cannot be construed to have an individual relation to the other elements. Likewise for $hurt'$ in the second case. The first one might arise from an expression such as *Mary thinks that love hurts*, and the second one from *Mary’s love hurt John*.

For example, in $sleep'kafka,'$ $sleep'$ depends on $kafka.'$ In $like'milena'kafka,'$ there are two dependency relations: $like'$ depends on $milena,'$ and $like'milena'$ depends on $kafka.'$ These embeddings follow from the left-associativity of juxtaposition, which we can show as:



The relation can be abstracted over. For example $\lambda x.sleep'x$ abstracts over the argument of a dependency, and $\lambda f.f.kafka'$ over the function.

We can take the tree above to signify the obliqueness of the arguments of the predicate in the prefix. We can say that a leaf node that c-commands another in the PADS is less oblique.¹¹¹ That is one of the reasons why we consider PADS to be a structure rather than a flat list. There would be no obliqueness relation for the leftmost element of a PADS. For example *slept'john'* does not manifest obliqueness. We can also say that a predicate “sees” its arguments one at a time in PADS: the elements that c-command the leftmost element in its PADS are its arguments. In the example below, the arguments of *p* are *a*, *(bc)* and *d*, not *a, b, c* and *d*.



We shall see in the book a combinatory equivalent of arity and argument structure specification, without the need of another primitive such as c-command. Order and its semantics will be doing the work, rather than auxiliary assumptions. Notice that we have already obtained the result from juxtaposition that obliqueness relations are asymmetries; no two arguments can c-command each other in the notation $pred'arg'_1arg'_2\cdots arg'_n$.

Notes

1. *Enjoy the silence*, Depeche Mode. Lyrics by Martin Gore.
2. Songbooks are not the right sources for Fraser words. They would be reinterpretations. Try a live performance or soundtrack of Cocteau Twins, with Liz Fraser singing her own words in the truest sense.
3. Dissemination by personal contact seems to be the fate of Schönfinkel's work. His other paper, the only other work he published, Schönfinkel (1929), was also prepared for publication by a colleague, Paul Bernays, who was in Göttingen at the time of the 1920 seminar. Curry's personal notes reveal that Bernays helped Behmann with the preparation of the 1924 article as well.
4. Besides the theory of Combinatory Categorical Grammar there is also the sub-field of Planning in Artificial Intelligence which makes heavy use of the semantics of adjacency, not to mention the most rapidly growing community in computing, the functional programming community including Lisp, Haskell, Javascript, Python, Ruby among many others. All of these make use of combinators. There is also a real computer architecture called SKIM (Clarke et al. 1980) in which the only primitive instructions are Schönfinkel's combinators.
5. For independent discovery and rediscovery of the principles involved, see Frege (1891), Quine (1966), de Bruijn (1972).
6. For brevity, I use *pefo'* as an abbreviation for the semantics of *persuade every friend of*. Likewise *tyf'* for *to vote for*. For the semantics of (7d) I follow Hoyt and Baldridge (2008): the '?' operator is variable-binding for the question *Q*. We shall see in the book that the research program of CCG is not to eliminate the variables such as *x* in this example, although it is certainly doable by combinators.
7. There is another way to make complex symbols out of simple values. Feature-based theories of syntax follow this path. For example, we can employ *subtyping* as in HPSG (Pollard and Sag 1994) to define e.g. *clausal-subject* and *nominal-subject* as subtypes of *subject* above. These types can be made part of a theory of features for a combinatory system too, as Beavers (2004) and McConville (2006) do. They are, however, different from the combinatory syntactic type in not having a strictly sequent semantics.
8. From a psycholinguistic perspective, the only recourse would be slips and false starts, in which some *new* types would be involved rather than a reinspection of the used types. Thanks to Belma Haznedar for clarification.
9. The entries in (9c–f) must be related because they arise from the same lexeme. This has to do with syntax-phonology-morphology interface and theory of the lexicon. These aspects are not covered in the book in detail.
10. It was Göksel (2006) who first observed the anaphoric behavior of the plural

- and possessive suffixes.
11. Coordination of unlike categories, such as *John is a republican and proud of it*, does not have true coordination semantics: **John is proud of it and a republican*. We should also be wary of an accidental capture of coordination by like-categories: *John bought a beer and drank it*, versus *John drank it and bought a beer*, which has a different meaning. This is an early warning that syntax alone cannot account for all semantic constraints. Jacobson's (1999) warning for the insufficiency of the like-category constraint for CSC and its across the board consequences arises from another semantic problem, that of reference, and it raises similar concerns for variable-friendly syntax and semantics; see Chapter 6.
 12. I write some of the left-hand sides in single quotes because, strictly speaking, they are not syntactic types in the combinatory sense; they can be thought of as the morphological sources of the syntactic types.
 13. I will follow in the book a common view that morphological types relate to form, syntactic types to constituency and semantic types to interpretation. The distinction is crucial for many theories such as the Separation Hypothesis in morphology (Beard 1995), in which morphological types do not "see" semantic types. Distributed Morphology (Halle and Marantz 1993) suggests that it is the phonological material that cannot see the other kinds of information because it is inserted after the syntactic process. As all theories agree that syntactic and semantic types must see each other to do compositional semantics, this issue is not too critical to combinatory syntactic types, which is our main focus.
 14. We are presently assuming that *love'* is not innately typed; otherwise we would know without exposure that it is $(e, (e, t))$ semantically. This knowledge is acquired. An implication of the present discussion to word acquisition is that the complete interpretability of words is an intrinsic part of their knowledge, rather than innateness of e.g. the transitive construction or some universal argument structure. This knowledge, which we might consider to be the syntactic reflex of the child's cognitive burden of attempting to make sense of the world, narrows down the search problem in language acquisition. The relation of the task to syntactic types is implicit in e.g. Siskind (1995). We shall see more detailed examples in §9.5.
 15. This view has been advocated much earlier, and it was severely underappreciated by the transformational grammarians. Halliday (1966, 1970), Halliday and Hasan (1976) had not claimed that there is no structure in language, in written or oral text, only that we should look for it where it really mattered, and where it can be observed to be at work.
 16. A variant of transformationalism such as that of Kayne (1994) in which order is seen as a reflex of structure might seem similar to CCG in comparison to other theories mentioned. The two programs are not compatible because CCG's conjecture amounts to saying that structure is a reflex of order, an opposite

conclusion with respect to Kayne. Likewise, Hawkins (2001)-style adjacency effects to explain the category adjacency rely on structural domains minimized on structural aspects, which presupposes structure-dependence rather than combinatory type-dependence. Notice also the crucial use of movement in Kayne's hypothesis to claim a universal subject-verb-object word order, which compromises the use of adjacency for syntax and semantics.

17. It was explicit in Curry's notation before he met Schönfinkel (1920/1924) in a literature search. Curry (1927) in his personal notes translates for example Schönfinkel's $\mathbf{I}a$ to his then-current notation $\mathbf{I}@a$. Curry (1929) notes that, in (xy) nothing is said if x is not a function, and suggests taking such (xy) to be equal to $\mathbf{K}xy$, which we will follow.
18. If f is binary, a purported definition such as $f(x_1)$ can be understood to be lossy only by investigating the "body" of f , which would make use of another argument, say x_2 . Similarly, $f(x_1, x_2, x_3, x_4)$ could be found to be too liberal if the body of f makes no use of say x_3 and x_4 . Both cases treat arity as an illative notion rather than a stipulative property of f . This may be a better way to proceed in cognitive science rather than the axiomatic approaches to argument-taking commonly assumed in computing and linguistics, provided that we can manage to keep infinite regress under control and stay empirically sound at the same time. For example, no language has manifested a ditransitive *sleep* predicate where only two arguments are syntactically available, and no language has a syntactically argumentless verb. These facts want explaining rather than stipulation.
19. Penrose's more famous conjecture, that the human mind is noncomputable, is not relevant here because a theory to predict possible languages would not be a theory to predict possible minds, unless of course we believe language is all there is to mind.
20. Chomsky (1965: 62): "[...] It is important to realize that the questions presently being studied are primarily determined by feasibility of mathematical study, and it is important not to confuse this with the question of empirical significance."
21. For our purposes, it suffices to note that the primitive recursive languages are languages of functions as programs which can be written without indefinite looping such as "repeat" or "while", and where the notion of "next instance" plays a crucial role. Not all recursive languages are primitive recursive, for example the Ackermann function.
22. Chomsky (1965: 208:fn.37): "This possibility [that the least powerful empirically adequate theory might turn out to be equivalent in weak or strong generative capacity to Turing machines] cannot be ruled out a priori, but in fact, it seems definitely not to be the case. In particular, it seems that, when the theory of transformational grammar is properly formulated, any such grammar must meet formal conditions that restrict it to the enumeration of recursive sets." Levelt (1974) is more explicit. He equates descriptive adequacy of a theory

with providing linguistic grammars that stay within recursive grammars, and explanatory adequacy with providing primitive recursive grammars, i.e. there must be a way to see how the grammar is caused. Thus for Levelt, any grammar for a natural language must be decidable.

23. Infinite regress is not a concern here because it can be avoided so long as we do not ask for the entire solution space at once. Consider a Putnam-Gold machine M_1 which takes another Putnam-Gold machine M_2 as input. M_1 can leave an initial answer on the result tape, and reconsider its output if similarly operating M_2 changes its output. Both M_1 and M_2 will have fetchable answers at any time, although they may both be undecidable. What we cannot have is M_1 to ask whether M_2 has stopped and delivered *all* its results. The process is reminiscent of *lazy evaluation* in programming languages, although they arise from different concerns.
24. In a nutshell, the most demanding task in the execution of a program is access to names. As most programming languages allow nested definition of names, the task is exacerbated by the look-up of names which are not local to the currently executing subprogram but defined elsewhere. The theory of compiling has found ingenious methods to tackle the problem. The problem becomes a nonproblem when there are no variables. With this in mind, programming language design and compiling become the art of translating a programmer's specification, which includes variables for the benefit of the programmer, to a variableless executable code.
25. The statement is attributed to Merrill Garrett by Fodor (1983). Chomsky (2000: 124) considers it problematic: "The belief that parsing is "easy and quick," in one familiar formula—and that the theory of language design must accommodate this fact—is erroneous; it is not a fact." He considers it to be a performance issue, and needlessly complicating a competence grammar since parsing according to him is not its business. It is not clear to me what Chomsky means by "design" in a product of evolution, but other conceptions of competence, such as Bresnan and Kaplan's (1982b) Strong Competence Hypothesis where the performance grammar just follows the instructions of the competence grammar, or Steedman's (2000b) Strict Competence Hypothesis where competence grammar is the performance grammar, take more burden of proof on their shoulders than Chomsky, by taking Garrett's remark as an empirical observation *about* grammar. This is essentially the view adopted in Levelt (1974: 236) as well: "The data for competence research are linguistic judgments, which are forms of language behavior. It is not clear why just this type of language behavior (linguistic judgment) should have the privilege of leading to a theory."
26. It should come as no surprise that one of the earliest objections to semantic vacuousness of some words is from one the most prominent semanticists and phonologists of the 20th century, Dwight Bolinger (1977).
27. Ades and Steedman (1982) and Szabolcsi (1983) appear to be the first syntacti-

- cizations of this kind. Geach (1972) is a syntacticization of composition as well, from the perspective of set theory, following Quine. Up until Steedman (1985, 1988), Szabolcsi (1987a), CCG developed independently of Schönfinkel's and Curry's combinators.
28. Note that $(A/B)/C: f$ is a two-argument function whereas $A/(B/C): f$ is a one-argument function.
 29. Smullyan pays homage to Schönfinkel and Curry in the choice of species as well. The book is dedicated to Curry, an avid bird-watcher. Smullyan (1985: 241) has his inspector Craig's trusted friend Fergusson cook up a story that Schönfinkel means "beautiful bird" in German. The Yiddish suffix "-el" adds a morphological mystery to ornithological logic.
 30. The first appearance of the paradoxical combinator in publication is Rosenbloom (1950), who called it Θ . Curry had worked on this combinator since 1929.
 31. Notice that the mismatch arises from the assumed sameness of semantics for X/Y and $X\backslash Y$, viz. b . As explained in the introduction, we can have $A \rightarrow C/B$ and $B \rightarrow C\backslash A$, if we know that A and B in this order derives C , i.e. $A B \Rightarrow C$. This equivalence spells the correspondence $X/Y: b \rightarrow X\backslash Y: \lambda a.ba$, which can arise from the configuration $Y: a X/Y: b \Rightarrow X: ba$.
 32. A system is called *applicative* if it uses application as the only primitive.
 33. The two combinators are obviously related. Curry and Feys (1958) give the following equivalence: $\Psi = \Phi(\Phi(\Phi\mathbf{B}))\mathbf{B}(\mathbf{K}\mathbf{K})$. The \mathbf{K} 's symbolize gapping.
 34. Smullyan's (1985) Eagle (\mathbf{E}) takes five arguments, like his Dickcissel and Dovekie, and they are the least visited birds in his book (also the seven-argument giant, the Bald Eagle).
 35. $\mathbf{Y} = \mathbf{SSK}(\mathbf{S}(\mathbf{K}(\mathbf{SS}(\mathbf{S}(\mathbf{SSK}))))\mathbf{K}$. It is cumbersome, but it does the job. We might go one step further and derive \mathbf{S} and \mathbf{K} from Barendregt's (1984) combinator \mathbf{X} , but not without some circularity. Take $\mathbf{X} = \lambda x.x\mathbf{KSK}$. Then $\mathbf{XXX} = \mathbf{K}$, and $\mathbf{X}(\mathbf{XX}) = \mathbf{S}$. The bottom line is, if we want the complete elimination of variables, we need the \mathbf{S} and \mathbf{K} somehow; witness \mathbf{KSK} in \mathbf{X} .
 36. Further optimizations are possible, for example using \mathbf{BCS} or $\mathbf{CD}\Phi$ to eliminate the unnecessary proliferation of \mathbf{S} abstractions; see Curry and Feys (1958: 188ff), Turner (1979).
 37. I suggest the name \mathbf{O} to symbolize its internalized lambda, and to acknowledge that it turned out to be different than \mathbf{D} in discussions with Umut Özge, Jason Baldrige and Frederick Hoyt. This combinator was named \mathbf{D} by Hoyt and Baldrige (2008) with the same semantics and syntax covered here. I proposed to change the name to avoid confusion with Rosenbloom's (1950) \mathbf{D} , which has different semantics and syntax.
 38. Take f to be $\lambda y.yb$, for some b . Then, for some a , we have $\lambda x.f(g(hx))a =_{\beta} g(ha)b$, but $f(\lambda x.g(hx))a =_{\beta} g(hb)a$.
 39. The theory began with Ades and Steedman (1982), written in 1979. Steedman

- developed the theory in a series of papers (Steedman 1985, 1987, 1988, 1990a,b, 1991a,b, 2000a). Synopses can be found in Steedman (1996b, 2000b), Steedman and Baldrige (2011).
40. On a historical note, the interdefinability of combinators was dealt with in a special section of Curry and Feys (1958), written by William Craig. Smullyan's (1985) engagement of a chief inspector of the same name to tackle ornithological affairs acknowledges this somewhat neglected contribution. In linguistics, interdefinability is prominent in Anna Szabolcsi's (1983, 1987b, 1989, 1992) work. She was principally involved in bringing **S** syntax to explanations, which was identified by Steedman to arise from **S** semantics.
 41. The idea was influential in the structure-dependent theories as well, starting with early transformations. It is most formally dealt with in Pollard (1984).
 42. Bach (1984: 7) defines the semantics of *persuade* in Montagovian terms: "*persuade* is interpreted as denoting a function from properties to a function from terms to sets". The property translates to a VP, and the function from terms to sets is a transitive verb, i.e. $(S \backslash NP) / NP$, hence the need for surface wrap.
 43. Sometimes the distinction is attributed to proof-theoretic versus model-theoretic approaches to syntax, but this is slightly misleading. It is true that CCG is a combinatory theory of adjacency syntax, rather than a set-theory of linguistic constraints. The words are the models though (assuming no words with **Y** semantics), because every constraint on a word's syntactic-semantic behavior must be reflected in its lexical category, hence any Montague-style valuation in a model frame can be reduced to truth conditions for sentences. Type-Logical Grammar leaves some proof-theoretic results, such as the provability of crossing compositions in CCG, to models.
 44. That is to say they are not Aristotelian categories. Husserl's categories are open-ended, and they do not rely on a set of basic categories determined *a priori*.
 45. Steedman (2000b: 54) defines these principles as follows. Consistency: "All syntactic combinatory rules must be consistent with the directionality of the principal function". Inheritance: "If the category that results from the application of a combinatory rule is a function category, then the slash defining directionality for a given argument in that category will be the same as the one(s) defining directionality for the corresponding argument(s) in the input function(s)."
 46. The claim here is that (20b) is ungrammatical with the intended coordination reading but fine as a parenthetical.
 47. See Baldrige (2002), Baldrige and Kruijff (2003), Beavers (2004), McConville (2006) for comprehensive attempts at a feature geometry for CCG.
 48. The stronger sense of radical lexicalization and its effects on constructions and constituency can be observed when we compare related grammar theories. Consider some cherished Construction Grammar examples below, quoted by Goldberg (1995) as part of the crucial data in her book's opening.

i. *I loaded the hay onto the truck.*

Anderson (1971)

ii. *I loaded the truck with the hay.*

Example (i) is claimed to semantically differ from (ii) over and above the meanings of the lexical items involved, where (ii) implies full loading in some sense, and (i) does not. No such difference seems to follow from the same construction with different lexical items (iii-iv):

iii. *I loaded the CD onto the multi-cd player.*

iv. *I loaded the multi-cd player with the CD.*

Moreover, we need to account for the following effect, where fullness or partialness of the readings seem to be restored across the board because of constituency:

v. *I loaded the truck with hay and the multi-cd player with CD.*

49. There is a prediction of CCG about this construction which awaits research. If the maximum arity in any lexicon is n , then the power of **B** must be bounded by $n-1$ to stay within the class of efficiently parsable linear-indexed grammars, therefore $n+1$ -sequent verbs of subordination is all it can handle. Steedman (2000b) suggests that $n=4$ for English. This issue brings back Shieber's (1985) warning that considering the possibility of bounded crossing reduces all linguistic arguments to finite structures. The book has already steered toward that direction by saying that something can be finite but vast, and we would still need a linguistic theory to sieve through possible structures. Following this route would not fall into the fallacy of turning to regular expressions as linguistic theories. The Kolmogorov-Chaitin complexity of describing all and only the possible structures with them would be prohibitive, and it would not amount to a theory. We would expect a theory to be much shorter than what it descriptively covers. Whether finite or infinite in their stringsets, the languages seem to manifest limited constituency and dependency. A language can be infinite in terms of its stringset but finite in terms of possible structures, as for example free operation in syntax (i.e. closure) might suggest. Given these aspects I consider the infinitude argument secondary in linguistic explanation.
50. Szabolcsi (1983) called it *connection*—recall Schönfinkel's name, *fusion*, for the same effect. Steedman (1988) related connection to **S**.
51. Szabolcsi (1989) might appear to introduce unary **B** to English syntax, but she does that only for syntactic objects, hence it is a lexicalization of unary **B**.
52. Having two categories for *dymuno* 'want' in (36–37) is empirically sound; the same differences can be observed in control verbs of other languages, for example English and Turkish: *The hair wants cutting*, and *Wittgenstein wants to like Russell*. They might arise from a single category of *want*,¹ but that is a matter of argument structure and the lexicon.

53. The ongoing discussion of observing the combinators' semantics in syntax must be distinguished from similarly inspired operator-based systems, i.e. systems which relate *two* expressions by the use of combinators, such as that of Shaumyan (1987). For example he notes that *That man, I hate him*, with the semantics $hate'xthatman'i'$, where x is presumably the pleonastic use engendered by *him*, is related to *I hate that man*, by **K**. Its semantics is $hate'thatman'i'$. I have nothing to say about such systems except to note that they need some notion of synonymy, and run into the same difficulties that face the *any*-debate on undecidability; see §3.3 for Hintikka's (1977) synonymy argument.
54. A point of clarification: a lexical rule in CCG means a unary rule that only refers to substantive—therefore lexical—categories. It does not mean a rule that gives us more lexical items.
55. This chapter arose from discussions with Umut Özge. Usual disclaimers apply.
56. Cf. fn. 24, where simplifying the use of bound variables for the benefit of the programmer is claimed to ease the task of software planning.
57. That discourse is perhaps necessarily involved in such examples is evidenced by the proposals that can provide their bound interpretation in syntax, such as that of Pinkal (1991: (12)) "A NP α can bind a pronoun β provided that β is in the c-command domain of the host quantifier of α 's discourse referent." Without an analysis of the English genitive, it is not clear how such examples might be accounted for by Jacobson's variable-free semantics.
58. The idea of type-raising all arguments in a grammar seems to go back to Montague (1973), Lambek (1958, 1961). Montague's set-theoretic type e is empty. His subjects must be $((e,t),t)$. Lambek's radical lexicalization translates all NP types in a phrase structure grammar to their grammatical roles, i.e. to their type-raised variety.
59. In a VSO language we cannot maintain in surface structure that the least oblique argument is the last one to combine. Keeping this as a universal was one motivation for Dowty (1996) to abandon the adjacency assumption of CCG and adopt a surface-wrap analysis.
60. The rule (28) has the same result semantics as z -NP, which can be verified from its configuration: $X/Z/Y: f \ Z: a \ Y/Z: g \rightarrow X: f(ga)a$, where $Z=NP$, and f has an inner semantic—lexical—wrap. Crucially, the rule avoids the unary **S** semantics of $\lambda g \lambda x. fx(gx)$, against which Jacobson (1999: 136) warns us to eliminate $His_{*i} \ mother \ loves \ every \ Englishman_i$. However, the rule (28) would produce $S/NP^{NP} \setminus NP_{3s}$ for *loves*, therefore it would derive *Mary loves him* wrongly, unless verb-medial languages by-pass the rule by some 'same directionality' constraint on the $|$'s, with predictable consequences for OSV languages such as Hixkaryana. Clearly there are restrictions on the syntactic type of f , ' $|_i$ ' and ' $|_j$ ' related to the crossover phenomena, which must remain currently as open questions.
61. For example, tag questions require a pronoun: *John will come, won't he?* Welsh

periphrastic passive requires a pronoun as an independent word. Steedman's model eschews the use of a distinct syntactic type for pronouns. Therefore in such constructions, the pronoun is predicted to be the head which can look for the arguments.

62. The degrees of freedom afforded by CCG in this domain is worth reiterating. As Steedman's (2011) LF eschews an exponent type in syntax, it cannot require it in a syntactic domain of locality. However, an analysis which takes the possessive pronoun as the head rather than *cael* is logically possible, and it will avoid an exponent type in the domains of locality. Such variations await research.
63. This lack of interaction between the parser stack and the quantifier store is most evident in the recent formulations of Cooper storage, such as in Pollard's (2008b) reworking of extended Montague Grammar to Convergent Grammar. His construal takes as its fundamental assumption the lack of an interaction.
64. To be more precise, there is no subject reflexive that can have an antecedent in the same clause. There are languages in which a subject "reflexive" can take an antecedent from a higher clause.
65. We can take the last sentence of the quote to suggest that the number of distinct PADS objects in a mental grammar is probably less than the number of syntactic objects, whereas the number of PADS tokens is probably higher, so that they are forced to recycle among the lexical entries to provide a network of relations. CCG is not designed to cope with such networks.
66. Notice that, if the string contains a syntactic displacement, say *the cat which I think sleeps is a menace*, where the substring '*sleeps is*' clearly does not embody an argumenthood relation between the two objects on its either side, the syntax of the other combinators involved will take care of the semantic dependencies to get the *sleep'* and *cat'* argumenthood right. The point of combinatory argument-encoding in a string of objects is that what cannot be torn apart and displaced separately is the \mathbf{B}^1 *sleep'* part, which comes from the lexicon.
67. a, b, g, h, i are from Baldrige (2002), c is from Steedman (2000b), and j is Steedman (p.c.). The use of $\mathbf{>O}_x$, $\mathbf{<O}_x$, $\mathbf{>S''}$ and $\mathbf{<S''}$ awaits further inquiry.
68. I take *en* to symbolize a syntactic feature such as $\mp en$, where *+en* is assumed for *cael*.

If we are told that semantically speaking the *cael* involved in the passive is not the same as active 'get', we can readjust our analysis to Jacobson-style pronouns and demand the passive *cael* to look for an NP^{NP} argument rather than *NP*.

69. The radical lexicalization of the passive using the possessive pronoun is further supported by the fact that 3sg form (ei) soft-mutates the uninflected verb, whereas 3pl (eu) does not (Awbery 1976:p.49). The analysis also coincides with Awbery's intuition that the phrase after *Wyn* in the example is a term of *cael*: notice the final dependency structure.
70. The across-the-board claim for the passive in languages of the world is that it is an operation which targets lexical verbs, and that that might be the reason why

it always targets the least oblique argument for demotion because every lexical verb has one. However, this line of reasoning does not explain why the passive promotes certain objects and not others. Our focus here is to work towards an explanation for its (clause) boundedness.

71. Lexical access to thematic structure does not in itself fully characterize the passive in relation to the reciprocal, causative and the reflexive. The property is proposed here as a necessary and insufficient condition to pin down the semantics of the passive.
72. There are exceptions in the verb-medial languages as well. For example, bare complements make it easier to break the word order constraint: *The cat which I knew (*that) would be a menace is Carlyle*; see Steedman (1996b) and Baldrige (2002) for extensive discussion.
73. Examples such as (18a) are sometimes considered ungrammatical by some Turkish syntacticians on the grounds that they are odd without a context. Since there is no such thing as null context, and because a competence grammar must provide a derivation no matter how unlikely a meaning is if it is grammatical, we must keep such examples on the agenda. To see that (18a) is grammatical, consider a case where the topic is Ahmet's strange shooting practices.
74. For example: use *S'* rather than *NP* if the argument is clausal, use an *NP* rather than *S'* if the semantics of the construction is participatory therefore lexically visible, as in the passive.
75. Some of the material in this section arose from discussions with Mark Steedman. I present here my recollection and conclusions. Possible misunderstandings are mine.
76. This is true of any kind of computation, not just CCG. For example, a common practice in programming language compiling is to replace tail recursion with simple iteration. This optimization cannot be done for nontail recursion, which would be the true reflection of **Y** in the syntax of a language.
77. Finitude is certainly not a mental block to creativity. Pullum and Scholz (2009) suggest that Japanese *haiku* compositions can continue forever because the possibilities are finite but vast: up to 10^{34} haikus, but certainly a lot less number due to other constraints, but still a vast number.
78. It is quite striking that the two philosophers who sharply differed from their precursors and contemporaries in ascribing to animals skills that are only different from humans in degree rather than in kind, Hume and Wittgenstein, essentially saw a continuous problem space for coordinated action and experience of living things. These fresh perspectives rightfully established them as the philosophers dearest to some cognitive scientists.
79. The parts of the lexicon that are not visible to syntactic processes are formal knowledge of words such as the word-formation rules of Anderson (1992), Aronoff (1994). They do not necessarily depend on syntactic types. A morphological theory must explain these processes by giving us a landscape of possible

- morphological types.
80. The question of coordination being asymmetrically sensitive to the left or right conjunct is dealt with in Steedman (2000b) from a CCG perspective.
 81. Examples (21–22) are from Özge and Bozsahin (2010).
 82. I assume that morphology-phonology at the interfaces handle *-en* versus *ge-V-en* alternation in Dutch passive morphology (including the choice of *-d* or *-t* in place of *-en*), and yield a morphemic segment which I symbolized as *-EN* above. In this process there would be no involvement of its syntactic category, assuming the Separation Hypothesis of Beard (1995). The syntactic types do the ordering of combination in the syntactic process. For example, the particle *op-* ‘up’ in *opgestegen* might be the source of telicity as van Hout (2000) suggests, and this would be carried over to syntax by the syntactic type of the lexical item which we can symbolize as *OP-*.
 83. For purists, we can assume that everything in the lexical conceptual structure is projected onto PADS but only a few members of the powerset is used by syntax, which shows the need for a theory of the lexicon although the powerset is in all likelihood finite.
 84. As Rey (1986) reminds us, another computationalist trend, strong AI, is similarly accused wrongly about its aspirations of computationalism, which is functionalism, not behaviorism.
 85. All CCG learners work within the parse-to-learn paradigm. The alternative, which is the learn-to-parse paradigm, seems inconsistent with Garrett’s observation reported earlier that parsing is a reflex; see Fodor (1998), Steedman and Hockenmaier (2007) for discussion. No-parse paradigm relies on lexical lookup of words, and it presumes a more or less disambiguated lexicon for the child, which does not seem very realistic.
 86. In earlier work (Çöltekin and Bozsahin 2007) we called β the likelihood. We thank Orkan Bayer for pointing out our error.
 87. This notion of “having a meaning” is not related to Quine’s use of the same term for grammars, as Chomsky never tires of pointing out; see e.g. Chomsky (2000: fn.18:199).
 88. To be more precise, Siskind’s cross-situational learning emphasizes the *likely* meanings of words rather than possible meanings, the latter of which Quine argued to be infinitely many. A similar narrowing of word meanings is defended from a linguistic perspective by Williams (1994).
 89. Much of Quine’s possible readings are eliminated by the parsimony principles of Siskind (1996). The list provided is only a first approximation for this process. For example, from Siskind’s principle of exclusivity, the child can conclude that chocolate does not mean whatever she assumed for *plu’* because in the first experience there is the plural assumption but no chocolate.
 90. I am grateful to Aravind Joshi for related discussion.
 91. Thanks to Alan Libert for these examples. I am responsible for the lexicalization

- claim.
92. There is something other than case marking and word order that comes to the rescue in the recovery of grammatical relations: agreement systems and noun classes; see Steele (1978), Mallinson and Blake (1981). Notice that, in a system of combinatory syntactic types, these morphological resources narrow down the syntactic types just like case marking, without different levels of structure or subsystems.
 93. There is a certain myth about scrambling languages. If asked in isolation, a speaker might say that a legitimately permuted sentence means more or less the same thing as the unpermuted ones. But this is hardly the right question. Provide theme or rheme alternatives before the example, and most speakers would prefer one word order only, if the alternatives are set to elicit that order. More interestingly, they would reject most of the others as either ungrammatical or contextually inappropriate, suggesting that there are other semantic reasons than who-does-what-to-whom.
 94. Here I assume a tripartite functional division of labor in parsing, following Steedman (2000b): (a) a grammar, (b) a parsing algorithm to derive strings using the grammar, and (c) an oracle to choose between the alternative derivations and potential ambiguities.
 95. These examples might be considered odd in a null context, but certainly not ungrammatical. They are perfectly interpretable in for example a partitive context in which there was a children's party where several delicacies were served, chocolate among them. I deliberately avoided the aorist *sever* 'loves' to rule out generic readings yet still maintain the indefinite ones. See Özge (2010) for more examples of indefinite accusatives, and for an argument that, in Turkish studies so far, pinning down the semantics of definiteness and specificity to the morphemes has not been very successful.
 96. A definite reading can be obtained in response to the question: *What did the kids think of the sweets we served?* The indefinite reading may follow from the question: *Can we say that we made all the guests happy?* The issue is unsettled; see Nakipoğlu (2009), Özge (2010) and references therein for extensive discussion.
 97. For an informative and entertaining exposure to monads and for their relation to interactions in computation, see Wadler (1997), who relates them to Descartes's mind-body problem.
 98. From the song *Flaming* in Pink Floyd's 1967 album, *The Piper at the Gates of Dawn*. Lyrics and music by Syd Barrett.
 99. This is similar to French *echaînement*, for example *faux ami* [fo][za][mi], but in the backward direction.
 100. The first two correspondences of (11) and the first one in (12) highlight an equivalence on the semantic side modulo eta-conversion of lambda calculus.
 101. It is explicit in any model of CCG that the bootstrapper for acquisition cannot

be just phonological or semantic; it must be grammatical because a lexicalized syntactic type is the only way to establish the correspondence of a string with some predicate-argument structure. See Steedman and Hockenmaier (2007) for discussion.

102. There is a combinatory equivalent of η 's ordered pair constructor (c, x) , which is based on \mathbf{D}_2 and \mathbf{Z}_n in Curry and Feys (1958). I eschew it here for shorter exposition.
 μ is commonly formulated as $\mathbf{BSC}fg = \lambda x.f(gx)x$ in the reader monads, as in Shan (2001), which in our case is $\mathbf{BSC}a(dU) = \lambda x.a(dUx)x$. The order of the dependencies (dUx) and x is not critical in the monad; order is already encoded in the input by η . \mathbf{BSC} does not encode a dependency which is functionally different than that of \mathbf{S} , hence my choice of the better-known combinatory term for μ in (14).
103. The monadic version coincides with Jacobson's (1999) use of composition as a sequence of unary \mathbf{B} followed by application, which is generalized in monadic grammar to apply to all combinators.
104. Another excursion of Curry to linguistics is Curry (1929), where he defends the grammarian's view of meaning over the logician's view of meaning.
105. This is unlike Locke's naive empiricism. We cannot assume *tabula rasa* for dependencies. And we must assume a syntactic specialization of combinators. Hume has always insisted that human beings bring something special to their understanding, and that they cannot help themselves attributing for example a causal link when there is no causation. In other words, some things are internalized to the point of a reflex.
106. We owe our current understanding of the Baldwin effect to Simpson (1953). Baldwin (1896) thought he had found a new cause for selection, which he called *organic selection*, in addition to natural selection. Simpson identified it to be an effect rather than a cause, and coined the name. The Simpsonian view of Baldwin is what makes Deacon's proposal tick. There seems to be coextensive but not separate mechanisms for selection.
107. It seems to be a major point for Bickerton and Chomsky (2000) that human evolution has more or less stopped—remember Chomsky's claim that language is a perfect system, and that only languages as phenotypes may contain imperfections. Anthropologists and biologists, not to mention evolutionary linguists and neuroscientists, consider that to be very unlikely; see Hawks versus Jones debate at Hawks (2008).
108. Strictly speaking, the Turing bird which Smullyan defined as \mathbf{U} would not be functionally equivalent to the Sage Bird \mathbf{Y} named after Curry. We need the equivalent of Turing's (1937) definition: $\mathbf{U}=(\lambda x\lambda y.y(xxy))(\lambda x\lambda y.y(xxy))$. From this we get $\mathbf{U}f=[(\lambda x\lambda y.y(xxy))(\lambda x\lambda y.y(xxy))]f$, which is equivalent to $f([\lambda x\lambda y.y(xxy)][\lambda x\lambda y.y(xxy)]f)$. It gives us a fixpoint combinator: $\mathbf{U}f = f(\mathbf{U}f)$. Like \mathbf{Y} , \mathbf{U} is infinitely typeable. Unlike \mathbf{Y} , \mathbf{U} is a supercombinator.

109. Smullyan might have been persuaded to call **Y** *Yeşilbaş*, Turkish for green duck (literally ‘green head’), rather than the hapless Sage bird. The common confusion about whether ducks are birds or birds are ducks—since we know that geese and ostriches *are* ducks—seems fertile ground to breed recursion the paradoxical way.
110. This is of course true in programming as well. Programmers will remember the bitter experience of writing recursive programs without base cases, or with base cases that are not reachable.
111. A simple version of c-command suffices for our purposes: x c-commands y in a structure if x does not dominate y , and the node immediately dominating x also dominates y .

Bibliography

- Abney, Steven
1987 The English noun phrase in its sentential aspect. Ph.D. diss., MIT, Cambridge, MA.
- Ades, Anthony E. and Mark Steedman
1982 On the order of words. *Linguistics and Philosophy* 4: 517–558.
- Aissen, Judith
1990 Towards a theory of agreement controllers. In *Studies in Relational Grammar 3*, Paul M. Postal and Brian D. Joseph (eds.), 279–320. University of Chicago Press.
- Ajdukiewicz, Kazimierz
1935 Die syntaktische konnexitat. *Studia Philosophica* 1: 1–27. English translation in S. McCall (ed): *Polish Logic*, Oxford University Press, 1967.
- Aksu-Koc, Ayhan A. and Dan I. Slobin
1985 The acquisition of Turkish. In *The Crosslinguistic Study of Language Acquisition, vol. I: The Data*, Dan I. Slobin (ed.). New Jersey: Lawrence Erlbaum.
- Anderson, Stephen R.
1971 On the role of deep structure in semantic interpretation. *Foundations of Language* 6: 197–219.
1985 Typological distinctions in word formation. In *Language Typology and Syntactic Description*, Timothy Shopen (ed.), vol. III, Grammatical categories in the lexicon, 3–56. Cambridge: Cambridge University Press.
1992 *A-Morphous Morphology*. Cambridge Univ. Press.
- Aronoff, Mark
1994 *Morphology by Itself: Stems and Inflectional Classes*. Cambridge, MA: MIT Press.
- Awbery, G.M.
1976 *The Syntax of Welsh*. Cambridge Univ. Press.
- Bach, Emmon
1976 An extension of classical transformational grammar. In *Problems in Linguistic Metatheory: Proceedings of the 1976 Conference at Michigan State University*, 183–224. Lansing: Michigan State University.
1980 In defense of passive. *Linguistics and Philosophy* 3: 297–341.
1984 Some generalizations of categorial grammars. In *Varieties of Formal Semantics*, Fred Landman and Frank Veltman (eds.). Dordrecht: Foris.

- Baker, Mark C.
 2001 *The Atoms of Language*. New York: Basic Books.
- Baldrige, Jason
 2002 Lexically specified derivational control in Combinatory Categorical Grammar. Ph.D. diss., University of Edinburgh.
- Baldrige, Jason and Geert-Jan Kruijff
 2003 Multi-modal Combinatory Categorical Grammar. In *Proceedings of 11th Annual Meeting of the European Association for Computational Linguistics*, 211–218. Budapest.
- Baldwin, James Mark
 1896 A new factor in evolution. *The American Naturalist* 30: 441–451, 536–553.
- Bar-Hillel, Yehoshua, Chaim Gaifman and Eliyahu Shamir
 1960 On categorial and phrase structure grammars. *The Bulletin of the Research Council of Israel* 9F: 1–16.
- Barendregt, Henk P.
 1984 *The Lambda Calculus—Its Syntax and Semantics*. North-Holland. 2nd ed.
- Barker, Chris and Pauline Jacobson
 2007 Introduction: Direct compositionality. In *Direct compositionality*, Chris Barker and Pauline Jacobson (eds.), 1–19. Oxford: Oxford University Press.
- Barker, Chris and James Pryor
 2010 Seminar in semantics/philosophy of language. Lecture notes, New York University.
- Batman-Ratyosyan, Natalie and Karin Stromswold
 1999 What Turkish acquisition tells us about underlying word order and scrambling. *U. Penn Working papers in Linguistics* 6 (1): 37–52.
- Beard, Robert
 1987 Morpheme order in a lexeme/morpheme based morphology. *Lingua* 72: 73–116.
 1995 *Lexeme-Morpheme Base Morphology*. Albany, NY: SUNY Press.
- Beavers, John
 2004 Type-inherited Combinatory Categorical Grammar. In *Proc. of the 20th COLING*. Geneva.
- Berwick, Robert and Amy Weinberg
 1982 Parsing efficiency, computational complexity, and the evaluation of grammatical theories. *Linguistic Inquiry* 13: 165–192.
- Bickerton, Derek
 1990 *Language and Species*. University of Chicago Press.
 1996 *Language and Human Behavior*. University of Washington Press.

- Bickhard, Mark H.
 1996 Troubles with computationalism. In *Philosophy of Psychology*, W. O'Donohue and R. Kitchener (eds.), 173–183. London: Sage.
- Bird, Steven and T. Mark Ellison
 1994 One-level phonology: Autosegmental representations and rules as finite automata. *Computational Linguistics* 20 (1): 55–90.
- Blake, Barry J.
 1990 *Relational Grammar*. London: Routledge.
- Bolinger, Dwight
 1968 *Aspects of language*. New York: Hartcourt, Brace and World.
 1977 *Meaning and Form*. London: Longman.
- Borsley, Robert, Maggie Tallerman and David Willis
 2007 *The Syntax of Welsh*. Cambridge: Cambridge University Press.
- Bos, Johan, Stephen Clark, Mark Steedman, James R. Curran and Julia Hockenmaier
 2004 Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, Geneva, 1240–1246. ACL.
- Bozsahin, Cem
 1998 Deriving the predicate-argument structure for a free word order language. In *Proceedings of COLING-ACL 1998*. Montreal.
 2002 The combinatory morphemic lexicon. *Computational Linguistics* 28 (2): 145–176.
- Bozsahin, Cem and Nicholas V. Findler
 1992 Memory-based hypothesis formation: Heuristic learning of common-sense causal relations from text. *Cognitive Science* 16 (4): 431–454.
- Brent, Michael R.
 1993 From grammar to lexicon: unsupervised learning of lexical syntax. *Computational Linguistics* 19 (2): 243–262.
- Bresnan, Joan
 1978 A realistic transformational grammar. In *Linguistic Structure and Psychological Reality*, Morris Halle, Joan Bresnan and George Miller (eds.), 1–59. Cambridge, MA: MIT Press.
- Bresnan, Joan and Ronald Kaplan
 1982a Introduction: Grammars as mental representations of language. In *The Mental Representation of Grammatical Relations*, Joan Bresnan (ed.), xvii–lii. Cambridge, MA: MIT Press.
 1982b *The Mental Representation of Grammatical Relations*. Cambridge, MA: MIT Press.
- Brody, Michael
 1995 *Lexico-Logical Form: A Radically Minimalist Theory*. Cambridge, MA: MIT Press.

- Brown, Penelope
 1998 Children's first verbs in Tzeltal: Evidence for an early verb category. *Linguistics* 36 (4): 713–753.
- Calder, Jonathan, Ewan Klein and Henk Zeevat
 1988 Unification categorial grammar. In *Proceedings of the 12th International Conference on Computational Linguistics*. Budapest.
- Carleton, Lawrence R.
 1984 Programs, language understanding, and Searle. *Synthese* 59 (2): 219–230.
- Carlson, Greg
 1977 Reference to kinds in English. Ph.D. diss., University of Massachusetts, Amherst.
- Carpenter, Bob
 1997 *Type-Logical Semantics*. Cambridge, MA: MIT Press.
- Çakıcı, Ruken
 2008 Wide-coverage parsing for Turkish. Ph.D. diss., University of Edinburgh.
- Çöltekin, Çağrı and Cem Bozsahin
 2007 Syllable-based and morpheme-based models of Bayesian word grammar learning from CHILDES database. In *Proc. of the 29th Annual Meeting of the Cognitive Science Society*. Nashville, TN.
- Chomsky, Noam
 1957 *Syntactic Structures*. The Hague: Mouton.
 1961 On the notion "rule of grammar". In *Structure of Language and Its Mathematical Aspects*, Roman Jakobson (ed.), 6–24. American Mathematical Society. Proceedings of Symposia in Applied Mathematics, vol. XII.
 1965 *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.
 1966 *Cartesian Linguistics*. New York: Harper and Row.
 1970 Remarks on nominalization. In *Readings in English Transformational Grammar*, R. Jacobs and P. Rosenbaum (eds.), 184–221. Waltham, Mass.: Ginn.
 1972 Some empirical issues in the theory of transformational grammar. In *Goals of Linguistic Theory*, Stanley Peters (ed.). Englewood Cliffs, NJ: Prentice-Hall.
 1975 *The Logical Structure of Linguistic Theory*. Chicago: University of Chicago Press.
 1976 Conditions on rules of grammar. *Linguistic Analysis* 2: 303–351.
 1981 *Lectures on Government and Binding*. Dordrecht: Foris.
 1993 A minimalist program for linguistic theory. In *The View from Building 20*, Kenneth Hale and Samuel Jay Keyser (eds.), 1–52. Cambridge, MA: MIT Press.

- 1995 *The Minimalist Program*. Cambridge, MA: MIT Press.
- 2000 *New Horizons in the Study of Mind and Language*. Cambridge: Cambridge University Press.
- 2001 Derivation by phase. In *Ken Hale: a Life in Language*, Michael Kenstowicz (ed.), 1–52. Cambridge MA: MIT Press.
- 2005 Three factors in language design. *Linguistic Inquiry* 36 (1): 1–22.
- Chomsky, Noam and George A. Miller
- 1963 Introduction to the formal analysis of natural language. In *Handbook of Mathematical Psychology*, R. Duncan Luce, Robert Bush and Eugene Galanter (eds.), vol. 2, 269–322. New York: Wiley.
- Church, Alonzo
- 1936 An unsolvable problem of elementary number theory. *American Journal of Mathematics* 58: 345–63.
- 1940 A formulation of the simple theory of types. *Journal of Symbolic Logic* 5: 56–68.
- Church, Alonzo and J. Barkley Rosser
- 1936 Some properties of conversion. *Transactions of the American Mathematical Society* 39 (3): 472–82.
- Clark, Stephen
- 1997 Binding and control in categorial grammar. Master's thesis, University of Manchester.
- Clark, Stephen and James R. Curran
- 2007 Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 33 (4): 493–552.
- Clarke, T. J.W., P. J.S. Gladstone, C. D. MacLean and A. C. Norman
- 1980 SKIM - the S, K, I reduction machine. In *Proceedings of the 1980 ACM conference on LISP and functional programming*, 128–135. (LFP '80) New York, NY, USA: ACM.
- Cooper, Robin
- 1983 *Quantification and Syntactic Theory*. Dordrecht: Reidel.
- Crain, Stephen and Paul Pietroski
- 2001 Nature, nurture and universal grammar. *Linguistics and Philosophy* 24: 139–186.
- Creider, Chet, Jorge Hankamer and Derick Wood
- 1995 Preset two-head automata and natural language morphology. *International Journal of Computer Mathematics* 58: 1–18.
- Croft, William
- 2001 *Radical Construction Grammar: Syntactic Theory in Typological Perspective*. Oxford: Oxford University Press.
- Curry, Haskell B.
- 1927 Notes on Schönfinkel. Curry archives.

- 1929 An analysis of logical substitution. *American Journal of Mathematics* 51: 363–384.
- 1961 Some logical aspects of grammatical structure. In *Structure of Language and Its Mathematical Aspects*, Roman Jakobson (ed.), 56–68. American Mathematical Society. Proceedings of Symposia in Applied Mathematics, vol. XII.
- 1963 *Foundations of Mathematical Logic*. McGraw-Hill.
- Curry, Haskell B. and Robert Feys
- 1958 *Combinatory Logic*. Amsterdam: North-Holland.
- De Beule, J., B. De Vylder and T. Belpaeme
- 2006 A cross-situational learning algorithm for damping homonymy in the guessing game. In *Artificial Life X: Proc. of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, 466–472.
- de Bruijn, N.G.
- 1972 Lambda calculus notation with nameless dummies. *Indagationes Mathematicae* 34: 381–92.
- Deacon, Terrence
- 1988 Human brain evolution I: Evolution of human language circuits. In *Intelligence and Evolutionary Biology*, H. Jerison and I. Jerison (eds.). Berlin: Springer-Verlag.
- 1997 *The Symbolic Species*. New York: Norton.
- Dennett, Daniel C.
- 1995 *Darwin's Dangerous Idea: Evolution and the Meanings of Life*. New York: Simon and Schuster.
- Derbyshire, Desmond
- 1979 *Hixkaryana*. (Lingua Descriptive Studies) Amsterdam: North-Holland.
- Dewdney, A. K.
- 1984 On the spaghetti computer and other analog gadgets for problem solving. *Scientific American* 250 (6): 19–26.
- Di Sciullo, Anna Maria and Edwin Williams
- 1987 *On the Definition of Word*. Cambridge, MA: MIT Press.
- Dixon, R.M.W.
- 1972 *The Dyrbal Language of North Queensland*. Cambridge: Cambridge University Press.
- Dowty, David
- 1996 Non-constituent coordination, wrapping, and Multimodal Categorical Grammars. In *International Congress of Logic, Methodology, and Philosophy*. Florence. August.
- Dowty, David, Robert Wall and Stanley Peters
- 1981 *Introduction to Montague Semantics*. Dordrecht: Reidel.

- Dromi, Esther
1987 *Early Lexical Development*. Cambridge: Cambridge University Press.
- Eisner, Jason
1996 Efficient normal-form parsing for Combinatory Categorical Grammar. In *Proceedings of the 34th Annual Meeting of the ACL*, 79–86.
- Ekmekçi, Fatma
1986 Significance of word order in the acquisition of Turkish. In *Studies in Turkish Linguistics*, D.J. Slobin and K. Zimmer (eds.), 253–264.
- Elman, Jeffrey
1990 Finding structure in time. *Cognitive Science* 14: 179–211.
- Epstein, Samuel D., Erich Groat, Ruriko Kawashima and Hisatsugu Kitahara
1998 *A Derivational Approach to Syntactic Relations*. Oxford: Oxford University Press.
- Eryılmaz, Kerem and Cem Bozsahin
2012 Lexical redundancy, naming game and self-constrained synonymy. In *Proc. of the 34th Annual Meeting of the Cognitive Science Society*. Sapporo, Japan.
- Eryiğit, Gülşen, Joakim Nivre and Kemal Oflazer
2008 Dependency parsing of Turkish. *Computational Linguistics* 34 (3): 357–389.
- Everett, Daniel L.
2005 Cultural constraints on grammar and cognition in Pirahã. *Current Anthropology* 46 (4): 621–646.
2009 Pirahã culture and grammar: A response to some criticisms. *Language* To appear.
- Fazly, Afsaneh, Afra Alishahi and Suzanne Stevenson
2010 A probabilistic computational model of cross-situational word learning. *Cognitive Science* 34: 1017–1063.
- Feldman, Jerome
2010 Embodied language, best-fit analysis, and formal compositionality. *Physics of Life Reviews* Target article.
- Filinski, Andrzej
1999 Representing layered monads. In *Proc. of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 175–188. San Antonio, Texas.
- Fodor, Janet Dean
1998 Parsing to learn. *Journal of Psycholinguistic research* 27 (3): 339–374.
- Fodor, Jerry
1983 *The Modularity of Mind*. Cambridge, MA: MIT Press.

Frege, Gottlob

1891 Function and concept. In *Translations from the Philosophical Writing of Gottlob Frege*, Peter Geach and Max Black (eds.). Oxford: Blackwell. 1966.

1893 *Grundgesetze der Arithmetik, Band I*. Jena: Verlag Hermann Pohle.

1904 What is a function? In *Translations from the Philosophical Writing of Gottlob Frege*, Peter Geach and Max Black (eds.). Oxford: Blackwell. 1966.

Garey, M.R. and D.S. Johnson

1979 *Computers and Intractability: A guide to NP-Completeness*. San Francisco: W.H. Freeman.

Gazdar, Gerald

1981 Unbounded dependencies and coordinate structure. *Linguistic Inquiry* 12: 155–184.

1988 Applicability of indexed grammars to natural languages. In *Natural Language Parsing and Linguistic Theories*, Uwe Reyle and Christian Rohrer (eds.), 69–94. Dordrecht: Reidel.

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum and Ivan Sag

1985 *Generalized Phrase Structure Grammar*. Oxford: Blackwell.

Geach, Peter

1972 A program for syntax. In *Semantics of Natural Language*, Donald Davidson and Gilbert Harman (eds.). Dordrecht: D. Reidel.

Gentner, Dedre

1982 Why nouns are learned before verbs: Linguistic relativity versus natural partitioning. In *Language Development, vol.2: Language, Thought and Culture*, Stan A. Kuczaj II (ed.), 301–334. Hillsdale, New Jersey: Lawrence Erlbaum.

George, L.M. and Jaklin Kornfilt

1981 Finiteness and boundedness in Turkish. In *Binding and Filtering*, F. Heny (ed.), 105–127. Cambridge, MA: MIT Press.

Gibson, Edward and Gregory Hickok

1993 Sentence processing with empty categories. *Language and Cognitive Processes* 8: 147–161.

Gibson, James

1966 *The Senses Considered as Perceptual Systems*. Boston, MA: Houghton-Mifflin Co.

Göksel, Aslı

2006 Pronominal participles in Turkish and lexical integrity. *Lingue e Linguaggio* 5 (1): 105–125.

Gold, E. M.

1965 Limiting recursion. *J. Symbolic Logic* 30: 28–48.

- 1967 Language identification in the limit. *Information and Control* 16: 447–474.
- Goldberg, Adèle
1995 *Constructions: A Construction Grammar Approach to Argument Structure*. Chicago, IL: Chicago University Press.
- Gould, Stephen Jay and Richard C. Lewontin
1979 The spandrels of San Marco and the Panglossian paradigm: A critique of the adaptationist programme. *Proc. of the Royal Society of London* B205: 581–598.
- Grimshaw, Jane
2000 Locality and extended projection. In *Lexical Specification and Insertion*, Jane Barbara Grimshaw Peter Coopmans, Martin Everaert (ed.), 115–134. Amsterdam: John Benjamins.
- Groenendijk, J. and M. Stokhof
1997 Questions. In *Handbook of Logic and Language*, Johan van Benthem and Alice ter Meulen (eds.). Cambridge, MA: MIT Press.
- Haegeman, Liliane
1998 Elements of grammar. In *Elements of Grammar: Handbook of Generative Syntax*, Liliane Haegeman (ed.). Dordrecht: Kluwer.
- Halle, Morris and Alec Marantz
1993 Distributed morphology and the pieces of inflection. In *The View from Building 20: Essays in Linguistics in Honor of Sylvain Bromberger*, Kenneth Hale and Samuel Jay Keyser (eds.). Cambridge, MA: MIT Press.
- Halliday, Michael
1966 Lexis as a linguistic level. In *In Memory of J.R. Firth*, C.E. Bazell, J.C. Catford, M.A.K. Halliday and R.H. Robins (eds.), 148–62. Longman.
1970 Language structure and language function. In *New Horizons in Linguistics*, John Lyons (ed.), 140–165. Harmondsworth: Penguin.
1978 *Language as Social Semiotic*. London: Edward Arnold.
- Halliday, Michael and Ruqaiya Hasan
1976 *Cohesion in English*. Longman.
- Hankamer, Jorge
1989 Morphological parsing and the lexicon. In *Lexical Representation and Process*, W. Marslen-Wilson (ed.). Cambridge, MA: MIT Press.
- Harman, G.H.
1963 Generative grammars without transformation rules: A defense of phrase structure. *Language* 39: 597–616.
- Hauser, Marc, Noam Chomsky and W. Tecumseh Fitch
2002 The faculty of language: What is it, who has it, and how did it evolve? *Science* 298: 1569–1579.

- Hawkins, John A.
 1994 *A Performance Theory of Order and Constituency*. Cambridge: Cambridge University Press.
 2001 Why are categories adjacent? *J. Linguistics* 37: 1–34.
- Hawks, John
 2008 Weblog. <http://johnhawks.net/weblog/topics/evolution/selection/jones-evolution-stopping-2008.html>. October 10, 2008.
- Hays, David G.
 1964 Dependency theory: A formalism and some observations. *Language* 40: 511–525.
- Higginbotham, James
 1982 Comments on Hintikka’s paper. *Notre Dame Journal of Formal Logic* 23 (3): 263–271.
- Hintikka, Jaakko
 1977 Quantifiers in natural languages: some logical problems II. *Linguistics and Philosophy* 1: 153–172.
 1980 On the *any*-thesis and the methodology of linguistics. *Linguistics and Philosophy* 4: 101–122.
- Hockenmaier, Julia, Gann Bierner and Jason Baldridge
 2004 Extending the coverage of a CCG system. *Research on Language and Computation* 2: 165–208.
- Hockenmaier, Julia and Mark Steedman
 2007 CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics* 33 (3): 356–396.
- Hoeksema, Jack
 1985 *Categorial Morphology*. New York: Garland.
- Hoffman, Beryl
 1993 The formal consequence of using variables in CCG categories. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics, Columbus, OH*, 298–300. San Francisco, CA: Morgan Kaufmann.
 1995 The computational analysis of the syntax and interpretation of “free” word order in Turkish. Ph.D. diss., University of Pennsylvania.
- Hopcroft, John E. and Jeffrey D. Ullman
 1979 *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison-Wesley.
- Hopper, Paul J. and Sandra A. Thompson
 1980 Transitivity in grammar and discourse. *Language* 56: 251–299.
- Hornstein, Norbert
 1995 *Logical Form: From GB to Minimalism*. Oxford: Blackwell.

- Hoyt, Frederick M.
 2006 Negative concord and restructuring in Palestinian Arabic: A comparison of TAG and CCG analyses. In *Proc. of the 8th Int. Conference on TAG and Related Formalism*. Sydney.
- Hoyt, Frederick M. and Jason Baldrige
 2008 A logical basis for the **D** combinator and normal form in CCG. In *Proc. of the Annual Meeting of the ACL*. Columbus, OH.
- Hudson, Richard A.
 1984 *Word Grammar*. Oxford: Blackwell.
- Hughes, R.J.M.
 1984 The design and implementation of programming languages. Ph.D. diss., Oxford.
- Husserl, Edmund
 1900 *Logical Investigations*. New York: Humanities Press. 1970 trans. by J. N. Findlay [Original German edition, 1900-1901.].
- Hutton, Graham
 2007 *Programming in Haskell*. Cambridge Univ. Press.
- Huybregts, Riny
 1976 Overlapping dependencies in Dutch. *Utrecht Working Papers in Linguistics* 1: 24–65.
- Jackendoff, Ray
 1997 *The Architecture of the Language Faculty*. Cambridge, MA: MIT Press.
- Jackendoff, Ray and Stephen Pinker
 2005 The nature of the language faculty and its implications for language evolution. *Cognition* 97: 211–225.
- Jacobsen, W.H. Jr.
 1979 Why does Washo lack a passive? In *Ergativity*, Frans Plank (ed.). Academic Press.
- Jacobson, Pauline
 1990 Raising as function composition. *Linguistics and Philosophy* 13: 423–476.
 1992 Flexible categorial grammars: Questions and prospects. In *Formal Grammar*, Robert Levine (ed.), 129–167. Oxford: Oxford University Press.
 1996 The locality of interpretation: the case of binding and coordination. In *proceedings of the 6th Conference on Semantics and Linguistic Theory*. (Cornell Working Papers in Linguistics) Ithaca NY: Cornell University.
 1999 Towards a variable-free semantics. *Linguistics and Philosophy* 22: 117–184.

- 2002 The (dis)organization of the grammar: 25 years. *Linguistics and Philosophy* 25: 601–626.
- 2007 Direct compositionality and variable-free semantics: the case of “Principle B” effects. In *Direct compositionality*, Chris Barker and Pauline Jacobson (eds.). Oxford: Oxford University Press.
- Jaeggli, Osvaldo A.
 1986 Passive. *Linguistic Inquiry* 17 (4): 587–622.
- Johnson-Laird, Philip N.
 1983 *Mental Models*. Cambridge, MA: Harvard University Press.
- Joshi, Aravind
 1985 How much context-sensitivity is necessary for characterizing structural descriptions: Tree Adjoining Grammars. In *Natural Language Parsing*, David Dowty, Lauri Karttunen and Arnold Zwicky (eds.), 206–250. Cambridge: Cambridge University Press.
- Joshi, Aravind and Yves Schabes
 1992 Tree-adjoining grammars and lexicalized grammars. In *Definability and Recognizability of Sets of Trees*, Maurice Nivat and Andreas Podelski (eds.). Princeton, NJ: Elsevier.
- Joshi, Aravind, K. Vijay-Shanker and David Weir
 1991 The convergence of mildly context-sensitive formalisms. In *Foundational issues in Natural Language Processing*, Peter Sells, Stuart Shieber and Tom Wasow (eds.), 31–81. Cambridge, MA: MIT Press.
- Jusczyk, P. W., E. A. Hohne and M. Newsome
 1999 The beginnings of word segmentation by English learning infants. *Cognitive Psychology* 39: 159–207.
- Kaplan, Ron and Joan Bresnan
 1995 Lexical-functional grammar: A formal system for grammatical representation. In *Formal Issues in Lexical Functional Grammar*, Mary Dalrymple, Ronald Kaplan, John Maxwell and Annie Zaenen (eds.). Stanford, CA: CSLI Publications.
- Kaplan, Ron and Annie Zaenen
 1995 Long-distance dependencies, constituent structure, and functional uncertainty. In *Formal Issues in Lexical Functional Grammar*, Mary Dalrymple, Ronald Kaplan, John Maxwell and Annie Zaenen (eds.). Stanford, CA: CSLI Publications.
- Kaplan, Ronald M. and Martin Kay
 1994 Regular models of phonological rule systems. *Computational Linguistics* 20 (3): 331–78.
- Karttunen, Lauri
 1989 Radical lexicalism. In *Alternative Conceptions of Phrase Structure*, Mark Baltin and Anthony Kroch (eds.). Chicago: University of Chicago Press.

- Kay, Martin
 1985 Parsing in functional unification grammar. In *Natural Language Parsing*, David Dowty, Lauri Karttunen and Arnold Zwicky (eds.), 251–278. Cambridge: Cambridge University Press.
- Kayne, Richard
 1994 *The Antisymmetry of Syntax*. Cambridge, MA: MIT Press.
- Klein, Ewan and Ivan Sag
 1985 Type-driven translation. *Linguistics and Philosophy* 8: 163–201.
- Knight, Chris, Michael Studdert-Kennedy and James R. Hurford (eds.)
 2000 *The Evolutionary Emergence of Language*. Cambridge: Cambridge University Press.
- Komagata, Nobo
 1997 Efficient parsing for CCGs with generalized type-raised categories. In *Proceedings of the 5th International Workshop on Parsing Technologies, Boston MA*, 135–146. ACL/SIGPARSE.
 1999 Information structure in texts: A computational analysis of contextual appropriateness in English and Japanese. Ph.D. diss., University of Pennsylvania.
- Kornfilt, Jaklin
 1984 Case marking, agreement, and empty categories in Turkish. Ph.D. diss., Harvard University.
 2005 Asymmetries between pre-verbal and post-verbal scrambling in Turkish. In *The Free Word Order Phenomenon: Its Syntactic Sources and Diversity*, J. Sabel and M. Saito (eds.), 163–179. Berlin/New York: Mouton de Gruyter.
- Kruijff, Geert-Jan M. and Jason Baldrige
 2004 Generalizing dimensionality in Combinatory Categorical Grammar. In *Proceedings of the 20th COLING*. Geneva, Switzerland.
- Kuhlmann, Marco and Joakim Nivre
 2006 Mildly non-projective dependency structures. In *Proc. of COLING-ACL*, 507–514. Sydney.
- Kural, Murat
 1994 Postverbal constituents in Turkish. Ms, UCLA.
 1997 Postverbal constituents in Turkish and the Linear Correspondence Axiom. *Linguistic Inquiry* 28 (3): 498–519.
- Kwiatkowski, Tom, Luke Zettlemoyer, Sharon Goldwater and Mark Steedman
 2010 Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*. Cambridge, MA.
 2011 Lexical generalization in CCG grammar induction for semantic parsing. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*. Edinburgh.

Lambek, Joachim

- 1958 The mathematics of sentence structure. *American Mathematical Monthly* 65: 154–170.
- 1961 On the calculus of syntactic types. In *Structure of Language and Its Mathematical Aspects*, Roman Jakobson (ed.), 166–178. American Mathematical Society. Proceedings of Symposia in Applied Mathematics, vol. XII.
- 1988 Categorical and categorical grammars. In *Categorical Grammars and Natural Language Structures*, Richard T. Oehrle, Emmon Bach and Deirdre Wheeler (eds.), 297–317. Dordrecht: Reidel.

Levelt, Willem J.M.

- 1974 Formal grammars and the natural language user: A review. In *Formal Grammars in Linguistics and Psycholinguistics*, W.J.M. Levelt and A. Barnas (eds.). The Hague: Mouton.

Lewis, Harry R. and Christos H. Papadimitriou

- 1998 *Elements of the Theory of Computation*. New Jersey: Prentice-Hall.

Lieber, Rochelle

- 1980 On the organization of the lexicon. Ph.D. diss., MIT. Published by Indiana Univ. Linguistics Club, 1981.

Łukasiewicz, Jan

- 1929 *Elementy Logiki Matematycznej*. Warsaw: Pwn. English translation published by Pergamon Press and Pwn, 1963.

Machery, Edouard

- 2006 Two dogmas of neo-empiricism. *Philosophy Compass* 4 (1): 398–412.

Mallinson, Graham and Barry Blake

- 1981 *Language Typology*. Amsterdam: North Holland.

Manning, Christopher D.

- 1996 *Ergativity: Argument Structure and Grammatical Relations*. Stanford, CA: CSLI.

Marconi, Diego

- 1997 *Lexical Competence*. Cambridge, MA: MIT Press.

May, Robert

- 1977 The grammar of quantification. Ph.D. diss., MIT, Cambridge, MA.
- 1985 *Logical Form*. Cambridge, MA: MIT Press.

McCarthy, John J.

- 1981 A prosodic theory of nonconcatenative morphology. *Linguistic Inquiry* 12 (3): 373–418.

McConville, Mark

- 2006 An inheritance-based theory of the lexicon in Combinatory Categorical Grammar. Ph.D. diss., University of Edinburgh.

- McWhinnie, Brian
 2000 *The CHILDES Project: Tools for Analyzing Talk*. Mahwah NJ: Lawrence Erlbaum.
- Melnyk, Andrew
 1996 Searle's abstract argument against strong AI. *Synthese* 108: 391–419.
- Mel'čuk, Igor A.
 1988 *Dependency Syntax: Theory and Practice*. Albany, NY: State Univ. of New York Press.
- Moggi, Eugenio
 1991 Notions of computation and monads. *Information and Computation* 93 (1): 55–92.
- Montague, Richard
 1970 Universal grammar. *Theoria* 36: 373–398. Reprinted in Montague 1974, 222–246.
 1973 The proper treatment of quantification in ordinary English. In *Approaches to Natural Language*, J. Hintikka and P. Suppes (eds.). Dordrecht: D. Reidel.
 1974 *Formal Philosophy: Papers of Richard Montague*. New Haven, CT: Yale University Press. Richmond H. Thomason, ed.
- Moortgat, Michael
 1988a *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus*. Dordrecht: Foris.
 1988b Mixed composition and discontinuous dependencies. In *Categorial Grammars and Natural Language Structures*, Richard T. Oehrle, Emon Bach and Deirdre Wheeler (eds.). Dordrecht: D. Reidel.
- Moortgat, Michael and Richard T. Oehrle
 1994 Adjacency, dependency and order. In *Proceedings of the 9th Amsterdam Colloquium*.
- Morrill, Glyn V.
 1994 *Type Logical Grammar: Categorial Logic of Signs*. Dordrecht: Kluwer.
- Nakipoğlu, Mine
 2009 The semantics of the Turkish accusative marked definites and the relation between prosodic structure and information structure. *Lingua* 119 (9): 1253–80.
- Nevins, Andrew, David Pesetsky and Cilene Rodrigues
 2009 Pirahã exceptionality: A reassessment. *Language* 85 (2).
- Niv, Michael
 1994 A psycholinguistically motivated parser for CCG. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics. Las Cruces, NM*, 125–132. San Francisco, CA: Morgan Kaufmann.

- Oehrle, Richard T., Emmon Bach and Deirdre Wheeler eds.
 1988 *Categorial Grammars and Natural Language Structures*. Dordrecht: D. Reidel. Compilation of the meeting in Tucson, Arizona, June 1985.
- Özge, Umut
 2010 Information and grammar: A study of Turkish indefinites. Ph.D. diss., Middle East Technical University, Informatics Institute.
- Özge, Umut and Cem Bozsahin
 2010 Intonation in the grammar of Turkish. *Lingua* 120: 132–175.
- Pareschi, Remo and Mark Steedman
 1987 A lazy way to chart-parse with categorial grammars. In *Proceedings of the 25th Annual Meeting of the ACL*, 81–88.
- Partee, Barbara H. and Mats Rooth
 1983 Generalized conjunction and type ambiguity. In *Meaning, Use, and Interpretation of Language*, Rainer Bauerle, Christoph Schwarze and Arnim von Stechow (eds.). Berlin: de Gruyter.
- Payne, Thomas E.
 1997 *Describing Morphosyntax*. Cambridge: Cambridge Univ. Press.
- Peirce, Charles Sanders
 1870 Description of a notation for the logic of relatives, resulting from an amplification of the conceptions of Boole's calculus of logic. *Memoirs of the American Academy of Sciences* 9: 317–78.
- Perlmutter, David M.
 1983 Personal vs. impersonal constructions. *Natural Language and Linguistic Theory* 1: 141–200.
- Pesetsky, David
 1985 Morphology and logical form. *Linguistic Inquiry* 16 (2): 193–246.
 1995 *Zero Syntax*. Cambridge, MA: MIT Press.
- Peters, Stanley and Robert Ritchie
 1973 On the generative power of transformational grammars. *Information Science* 6: 49–83.
- Peyton Jones, Simon L.
 1987 *The Implementation of Functional Programming Languages*. New York: Prentice-Hall.
- Phillips, Colin
 2003 Linear order and constituency. *Linguistic Inquiry* 34: 37–90.
- Pickering, Martin
 1993 Direct association and sentence processing: A reply to Gorrell and to Gibson and Hickok. *Language and Cognitive Processes* 8: 168–196.
- Pickering, Martin and Guy Barry
 1991 Sentence processing without empty categories. *Language and Cognitive Processes* 6: 229–259.

- 1993 Dependency categorial grammar and coordination. *Linguistics* 31: 855–902.
- Pierrehumbert, Janet and Julia Hirschberg
 1990 The meaning of intonational contours in the interpretation of discourse. In *Intentions in Communication*, Philip Cohen, Jerry Morgan and Martha Pollack (eds.), 271–312. Cambridge, MA: MIT Press.
- Pinkal, Manfred
 1991 On the syntactic-semantic analysis of bound anaphora. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 45–50. Berlin.
- Pollard, Carl
 1984 Generalized phrase structure grammars, head grammars, and natural languages. Ph.D. diss., Stanford University.
 2008a Convergent grammar. Lecture notes, 13th ESSLLI, Hamburg.
 2008b Cooper storage cures the common cold. NaTAL Workshop, Semantics and Inference, Nancy.
- Pollard, Carl and Ivan Sag
 1987 *Information-Based Syntax and Semantics, Vol. 1*. Stanford, CA: CSLI Publications.
 1994 *Head-driven Phrase Structure Grammar*. Chicago: University of Chicago Press.
- Pollock, Jean-Yves
 1989 Verb movement, UG and the structure of IP. *Linguistic Inquiry* 20: 365–424.
- Postal, Paul M. and John Robert Ross
 2009 Inverse reflexives. In *Time and Again: Theoretical Perspectives on Formal Linguistics: in Honor of D. Terrence Langendoen*, William D. Lewis, Simin Karimi, Heidi Harley and Scott O. Farrar (eds.). Amsterdam: John Benjamins. Linguistik Aktuell.
- Prevost, Scott
 1995 A semantics of contrast and information structure for specifying intonation in spoken language generation. Ph.D. diss., University of Pennsylvania.
- Pullum, Geoffrey K. and Barbara Scholz
 2009 Recursion and the infinitude claim. In *Recursion in Human Language*, Harry van der Hulst (ed.). Mouton de Gruyter.
- Pustejovsky, James
 1991 The generative lexicon. *Computational Linguistics* 17 (4): 409–441.
- Putnam, Hilary
 1961 Some issues in the theory of grammar. In *Structure of Language and Its Mathematical Aspects*, Roman Jakobson (ed.), 6–24. American

- Mathematical Society. Proceedings of Symposia in Applied Mathematics, vol. XII.
- 1965 Trial and error predicates and the solution of a problem of Mostowski. *J. Symbolic Logic* 30: 49–57.
- Quine, Willard van Orman
- 1951 Two dogmas of empiricism. *The Philosophical Review* 60: 20–43.
- 1960 *Word and Object*. Cambridge MA: MIT Press.
- 1966 Variables explained away. In *Selected Logic Papers*. New York: Random House.
- 1967 Commentary on Schönfinkel 1924. In *From Frege to Gödel*, Jean van Heijenoort (ed.). Cambridge, MA: Harvard Univ. Press.
- Rey, Georges
- 1986 What's really going on in Searle's "Chinese room". *Philosophical Studies* 50: 169–185.
- Rosenbloom, Paul
- 1950 *The Elements of Mathematical Logic*. New York: Dover Publications.
- Ross, John Robert
- 1967 Constraints on variables in syntax. Ph.D. diss., MIT. Published as *Infinite Syntax!*, Ablex, Norton, NJ, 1986.
- Rosser, J.B.
- 1935 A mathematical logic without variables. *Annals of Mathematics* 36: 127–150.
- Sandra, Dominiek
- 1998 What linguists can and can't tell about the human mind: A reply to Croft. *Cognitive Linguistics* 9: 361–378.
- Santelmann, Lynn M. and Peter W. Jusczyk
- 1998 Sensitivity to discontinuous dependencies in language learners: Evidence for limitations in processing space. *Cognition* 69 (2): 105–134.
- Schönfinkel, Moses Ilyich
- 1920/1924 On the building blocks of mathematical logic. In *From Frege to Gödel*, Jan van Heijenoort (ed.). Harvard University Press, 1967. Prepared first for publication by H. Behmann in 1924.
- 1929 Zum entscheidungsproblem der mathematischen logik. *Mathematische Annalen* 99: 342–372.
- Searle, John R.
- 1980 Minds, brains and programs. *The Behavioral and Brain Sciences* 3: 417–424.
- 1990a Is the brain's mind a computer program? *Scientific American* 262 (1): 26–31.
- 1990b Is the brain's mind a digital computer? *Proc. of American Philosophical Association* 64 (3): 21–37.

- 2001 Chinese Room argument. In *The MIT Encyclopedia of the Cognitive Sciences*, Robert A. Wilson and Frank C. Keil (eds.), 115–116. Cambridge, MA: MIT Press.
- Shan, Chung-Chieh
 2001 Monads for natural language semantics. In *Proc. of ESSLLI Student Session*, Kristina Striegnitz (ed.), 285–298. Folli.
- Shaumyan, Sebastian
 1977 *Applicational Grammar as a Semantic Theory of Natural Language*. Edinburgh University Press.
 1987 *A Semiotic Theory of Language*. Indiana University Press.
- Shi, R., A. Marquis and B. Gauthier
 2006 Segmentation and representation of function words in preverbal French-learning infants. In *Proc. of the 30th Annual Boston University Conference on Language Development*, D. Bamman, T. Magnitskaia and C. Zaller (eds.), 549–560. Somerville, MA: Cascadilla Press.
- Shieber, Stuart
 1985 Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8: 333–343.
 1986 *An Introduction to Unification-based Approaches to Grammar*. Stanford: CSLI.
- Simpson, George Gaylord
 1953 The Baldwin effect. *Evolution* 7: 110–117.
- Siskind, Jeffrey
 1995 Grounding language in perception. *Artificial Intelligence Review* 8: 371–391.
 1996 A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition* 61: 39–91.
- Slobin, Dan I. and Thomas G. Bever
 1982 Children use canonical sentence schemas: A crosslinguistic study of word order and inflections. *Cognition* 12: 229–265.
- Smith, A.D.M.
 2003 Intelligent meaning creation in a clumpy world helps communication. *Artificial Life* 9 (2): 175–190.
- Smullyan, Raymond
 1985 *To Mock a Mockingbird*. New York: Knopf.
- Stabler, Edward P.
 1997 Derivational minimalism. In *Logical Aspects of Computational Linguistics (LACL'96)*, Christian Retoré (ed.), 68–95. (Lecture Notes in Computer Science 1328) New York: Springer.

- 1999 Remnant movement and complexity. In *Constraints and Resources in Natural Language Syntax and Semantics*, Gosse Bouma, Erhard Hinrichs, Geert-Jan Kruijff and Dick Oehrle (eds.), 299–326. Stanford, CA: CSLI.

Steedman, Mark

- 1985 Dependency and coördination in the grammar of Dutch and English. *Language* 61 (3): 523–568.
- 1987 Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory* 5: 403–439.
- 1988 Combinators and grammars. In *Categorial Grammars and Natural Language Structures*, Richard T. Oehrle, Emmon Bach and Deirdre Wheeler (eds.). Dordrecht: D. Reidel.
- 1990a Constituency and coordination in a combinatory grammar. In *Alternative Conceptions of Phrase Structure*, Mark R. Baltin and Anthony S. Kroch (eds.). University of Chicago Press.
- 1990b Gapping as constituent coordination. *Linguistics and Philosophy* 13: 207–263.
- 1991a Structure and intonation. *Language* 67: 260–298.
- 1991b Type raising and directionality in combinatory grammar. In *Proceedings of the 29th Annual Meeting of the ACL*, 71–78.
- 1996a Does grammar make use of bound variables? In *Proc. of the Conf. on Variable-free Semantics*, Michael Böttner and Wolf Thümmel (eds.). Osnabrück.
- 1996b *Surface Structure and Interpretation*. Cambridge, MA: MIT Press.
- 1999 Quantifier scope alternation in CCG. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, College Park, MD*, 301–308. San Francisco, CA: Morgan Kaufmann.
- 2000a Information structure and the syntax-phonology interface. *Linguistic Inquiry* 31: 649–689.
- 2000b *The Syntactic Process*. Cambridge, MA: MIT Press.
- 2002 Plans, affordances, and combinatory grammar. *Linguistics and Philosophy* 25: 723–753.
- 2005a Grammar acquisition in child and machine. In *Proc. of the 9th Conf. on Computational Natural Language Learning*. Ann Arbor, MI.
- 2005b Interfaces and the grammar. In *Proceedings of the 24th West Coast Conference on Formal Linguistics, Vancouver, March 2005*, John Alderete et al. (ed.), 19–33. Somerville, MA: Cascadilla Proceedings Project.
- 2006 A Zipfian view of Greenberg 20. Ms., University of Edinburgh.
- 2009 Welsh syntactic soft mutation without movement or empty categories. Ms. Univ. of Edinburgh.
- 2011 *Taking Scope*. Cambridge, MA: MIT Press.

- Steedman, Mark and Jason Baldrige
 2011 Combinatory Categorical Grammar. In *Non-transformational syntax*, R. Borsley and Kirsti Börjars (eds.), 181–224. Oxford: Blackwell.
- Steedman, Mark and Julia Hockenmaier
 2007 The computational problem of natural language acquisition. Ms., University of Edinburgh.
- Steele, Susan
 1978 Word order variation: A typological study. In *Universals of Human Language*, Joseph Greenberg (ed.). Stanford University Press.
- Stoy, J.E.
 1981 *Denotational Semantics*. Cambridge, MA: MIT Press.
- Szabolcsi, Anna
 1983 ECP in Categorical Grammar. Ms., Max-Planck Institute.
 1987a Bound variables in syntax: Are there any? In *Proceedings of the 6th Amsterdam Colloquium*, 331–350.
 1987b On Combinatory Categorical Grammar. In *Proceedings of the Symposium on Logic and Language, Debrecen*, 151–162. Budapest: Akadémiai Kiadó.
 1989 Bound variables in syntax: Are there any? In *Semantics and Contextual Expression*, Renate Bartsch, Johan van Benthem and Peter van Emde Boas (eds.), 295–318. Dordrecht: Foris.
 1992 On combinatory grammar and projection from the lexicon. In *Lexical Matters*, Ivan Sag and Anna Szabolcsi (eds.), 241–268. Stanford, CA: CSLI Publications.
 1994 The noun phrase. In *The Syntactic Structure of Hungarian*, Ferenc Kiefer and Katalin É Kiss (eds.). (Syntax and semantics) San Diego: Academic Press.
 2003 Binding on the fly: Cross-sentential anaphora in variable-free semantics. In *Resource Sensitivity in Binding and Anaphora*, Geert-Jan Kruijff and Richard T. Oehrle (eds.), 215–229. Dordrecht: Kluwer.
- Tardif, Twila
 1996 Nouns are not always learned before verbs: Evidence from Mandarin speakers' early vocabularies. *Developmental Psychology* 32 (3): 497–504.
- Tesnière, Lucien
 1959 *Éléments de Syntaxe Structurale*. Paris: Editions Klincksieck.
- Thiessen, Erik D. and Jenny R. Saffran
 2003 When cues collide: Use of stress and statistical cues to word boundaries by 7- to 9-month-old infants. *Developmental Psychology* 39 (4): 706–716.

Trechsel, Frank

- 2000 A CCG account of Tzotzil pied piping. *Natural Language and Linguistic Theory* 18: 611–663.

Turing, Alan Mathison

- 1936 On computable numbers, with an application to the entscheidungsproblem. *Proc. of the London Mathematical Society* 42 (series 2): 230–265.
- 1937 Computability and λ -definability. *J. of Symbolic Logic* 2 (4): 153–163.
- 1950 Computing machinery and intelligence. *Mind* 59 (236): 433–460.

Turner, David A

- 1979 Another algorithm for bracket abstraction. *Journal of Symbolic Logic* 44: 267–270.

van Hout, Angeliek

- 2000 Event semantics in the lexicon-syntax interface. In *Events as Grammatical Objects*, Carol Tenny and James Pustejovsky (eds.), 239–282. Stanford: CSLI.

Vijay-Shanker, K. and David Weir

- 1990 Polynomial time parsing of Combinatory Categorical Grammars. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics, Pittsburgh*, 1–8. San Francisco, CA: Morgan Kaufmann.
- 1994 The equivalence of four extensions of context-free grammar. *Mathematical Systems Theory* 27: 511–546.

Villavicencio, Aline

- 2002 The acquisition of a unification-based generalised categorial grammar. Ph.D. diss., University of Cambridge.

Wadler, Philip

- 1990 Comprehending monads. In *Proc. of ACM Conference on Lisp and functional programming*.
- 1997 How to declare an imperative. *ACM Computing Surveys* 29 (3).

Williams, Edwin

- 1994 Remarks on lexical knowledge. *Lingua* 92: 7–34.

Wittgenstein, Ludwig

- 1942 *Blue and Brown Books*. London: Harper Perennial.

Yang, Charles

- 2006 *The Infinite Gift*. New York NY: Scribner.

Zaenen, Annie

- 1991 Subcategorization and pragmatics. Presentation at CSLI, Stanford.
- 1993 Unaccusativity in Dutch: Integrating syntax and lexical semantics. In *Semantics and the Lexicon*, James Pustejovsky (ed.). Kluwer.

Zettlemoyer, Luke and Michael Collins

- 2005 Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of the 21st Conf. on Uncertainty in Artificial Intelligence*. Edinburgh.
- 2007 Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, 678–687. ACL.

Zurif, Edgar B.

- 1995 Brain regions of relevance to syntactic processing. In *An Invitation to Cognitive Science: Language*, Lila R. Gleitman and Mark Liberman (eds.). MIT Press.

Author and name index

- Abney Steven, 165
Ackermann Wilhelm, 237
Ades Anthony, viii, 24, 58, 73, 205,
238, 239
Aissen Judith, 170
Ajdukiewicz Kazimierz, 5
Aksu-Koç Ayhan, 212
Allen James, 106
Anderson Stephen, 14, 81, 241, 244
Aronoff Mark, 12, 244
Awbery Gwen, 65, 79, 94, 99, 126,
197
- Bach Emmon, 59, 62, 63, 87, 121,
124, 205, 206, 240
Baker Mark, 211
Baldrige Jason, 4, 33, 57, 65, 68, 69,
89, 90, 93, 114, 115, 119,
125, 127, 148, 175, 184,
185, 198, 201, 235, 239,
240, 243, 244
Baldwin Mark, 247
Bar-Hillel Yehoshua, 9, 80, 81, 211
Barendregt Henk, 31, 43, 239
Barker Chris, 87, 95, 98, 195
Barrett Syd, 188, 208, 246
Barry Guy, 41, 114
Batman-Ratyosyan Natalie, 212
Beard Robert, 12, 236, 245
Beavers John, 235, 240
Behmann Heinrich, 1, 31, 235
Bernays Paul, 235
Berwick Robert, 225
Bickerton Derek, 210, 211, 247
Bickhard Mark, 149
Bird Steven, 207
Blake Barry, 105, 246
Bolinger Dwight, 39, 238
Borsley Robert, 97, 98
- Bos Johan, 157, 191
Bozsahin Cem, 119, 140, 144, 149,
150, 152–154, 157, 175,
187, 188, 191, 202, 212,
245
Brent Michael, 169
Bresnan Joan, 23, 24, 29, 39, 124, 238
Brody Michael, 28, 179
Brown Penelope, 155
- Çakıcı Ruken, 186
Çöltekin Çağrı, 150, 152–154, 157,
191, 202, 245
- Calder Jonathan, 70
Carleton Lawrence, 177
Carlson Greg, 3
Carnap Rudolf, 88
Carpenter Robert, 29, 65
Chaitin Gregory, 241
Chomsky Noam, ix, 18, 23, 24, 28,
36–38, 43, 58, 59, 70, 72,
88, 89, 98, 104, 124, 132,
133, 136, 137, 179, 180,
203, 205–207, 210, 211,
237, 238, 245, 247
- Church Alonzo, 31, 47, 81
Clark Stephen, 152, 171, 191
Clarke T., 235
Collins Michael, 150, 152, 153, 157,
160, 181, 191
Cooper Robin, 103
Craig William, 80, 239
Crain Stephen, 18, 81
Creider Chet, 63
Croft William, 208
Curran James, 152, 191
Curry Haskell, 1, 2, 31–33, 37, 44–46,
50, 52–56, 58, 79, 80, 87,
109, 133, 192, 194, 205–

- 207, 219, 221, 222, 235,
237, 239, 240, 247
- de Beul J., 149
de Bruijn N., 1, 235
Deacon Terrence, 210, 213
Dennett Daniel, 213
Derbyshire Desmond, 113
Dewdney A., 227, 228
DiScuillo Anna Maria, 157
Dixon R., 113
Dowty David, 11, 62, 242
Dromi Esther, 151
- Edelman Shimon, 157
Eisner Jason, 202
Ekmekçi Fatma, 212
Ellison Mark, 207
Elman Jeffrey, 150, 205
Epstein Samuel, 28, 207
Eryigit Gülşen, 186
Eryılmaz Kerem, 149
Everett Daniel, 132, 134–136
- Fazly Afsaneh, 157
Feldman Jerome, 106
Feys Robert, 31–33, 45, 46, 50, 52–
56, 80, 133, 192, 194, 219,
221, 222, 239, 240, 247
- Filinski Andrzej, 201
Findler Nicholas, 212
Fodor Janet, 245
Fodor Jerry, 238
Fraser Elizabeth, ix, 235
Frege Gottlob, 1, 31, 88, 235
- Garey M., 154, 155
Gazdar Gerald, 9, 21, 70, 73
Geach Peter, viii, 73, 91, 205, 239
Gentner Dedre, 150
George Leland, 165
Gibson Edward, 41
Gibson James, 157
- Goksel Asli, 235
Gold Mark, 37, 38, 143, 148, 152,
154, 240
Goldberg Adele, 208
Gore Martin, 235
Gould Stephen, 213
Grimshaw Jane, 164
Groenendijk J., 33
- Haegeman Liliane, 164
Halle Morris, 12, 206, 236
Halliday Michael, 122, 236
Hankamer Jorge, 63, 84
Harary Frank, 206
Harman G., 9
Hauser Marc, 37, 132–134, 136, 210
Hawkins John, 156, 237
Hawks John, 247
Hays David, 208
Herbrand Jacques, 1
Hickok Gregory, 41
Higginbotham James, 39
Hilbert David, 1
Hintikka Jaakko, 39, 242
Hirschberg Julia, 29, 140, 187
Hockenmaier Julia, 150, 152, 153,
157, 160, 185, 186, 191,
203, 211, 245, 247
Hockett Charles, 206
Hoeksema Jack, 13
Hoffman Beryl, 83, 119, 194
Hopcroft John, 226
Hopper Paul, 109
Hornstein Norbert, 203
Hoyt Frederick, 4, 33, 57, 65, 114,
115, 119, 201, 235, 239
Hudson Richard, 16, 208
Hughes R., 220
Humboldt Alexander von, 135
Hume David, vii, ix, 16, 38, 135, 152,
209, 212, 244, 247
Husserl Edmund, 65, 66, 68–70, 135,
177, 240

- Hutton Graham, 194
 Huybregts Riny, 73
- Jackendoff Ray, 109, 148, 210
 Jacobsen W., 124
 Jacobson Pauline, 56, 62, 71, 87–90,
 95, 97, 99–102, 108, 119,
 236, 242, 247
 Jaeggli Osvaldo, 125, 147
 Jakobson Roman, 206
 Johnson D., 154, 155
 Johnson-Laird Philip, 106
 Joshi Aravind, 40, 73
 Jusczyk Peter, 169
- Kaplan Ron, 29, 207, 238
 Karttunen Lauri, 66, 72
 Kay Martin, 70, 207
 Kayne Richard, ix, 156, 236
 Klein Ewan, 59
 Knight Chris, 210
 Kolmogorov Andrey, 241
 Komagata Nobo, 119, 202
 Kornfilt Jaklin, 165, 174
 Kruijff Geert-Jan, 65, 240
 Kuhlmann Marco, 208
 Kural Murat, 174
 Kwiatkowski Tom, 157
- Łukasiewicz Jan, 31
 Lambek Joachim, 72, 73, 193, 205–
 207, 242
 Levelt Willem, 38, 237, 238
 Lewis Harry, 226
 Lewontin Richard, 213
 Locke John, 135, 212, 247
- Machery Edouard, 212
 Mallinson Graham, 246
 Mandelbrot Benoit, 206
 Manning Christopher, 74, 76
 Marantz Alec, 12, 236
 Marconi Diego, 180, 181
- May Robert, 88
 McCarthy John, 207
 McConville Mark, 110, 119, 235, 240
 McIlroy Douglas, 183
 McWhinnie Brian, 157
 Melnyk Andrew, 181
 Mel'čuk Igor, 16, 208
 Moggi Eugenio, 195
 Montague Richard, ix, 9, 11, 59, 88,
 104, 105, 166, 205, 206,
 231, 240, 242, 243
 Moortgat Michael, 48, 65, 173
 Morrill Glyn, 65
- Nakipoğlu Mine, 246
 Nevins Andrew, 133
 Niv Michael, 202
 Nivre Joakim, 208
- Özge Umut, 140, 175, 176, 187, 188,
 245, 246
 Oehrle Richard, 65, 73
- Papadimitriou Christos, 226
 Pareschi Remo, 120
 Partee Barbara, 105, 106, 190
 Payne Thomas, 81, 123
 Peirce Charles, 1, 31
 Penrose Roger, 36, 237
 Perlmutter David, 170
 Pesetsky David, 88, 104
 Peters Stanley, 21, 36
 Peyton Jones Simon, 55, 221
 Phillips Colin, 24, 27
 Pickering Martin, 41, 114
 Pierrehumbert Janet, 29, 140, 187
 Pietroski Paul, 18, 81
 Pinkal Manfred, 242
 Pollard Carl, x, 23, 29, 70, 206, 235,
 240, 243
 Pollock Jean-Yves, 164
 Postal Paul, 104
 Prevost Scot, 119, 202

- Pryor James, 95, 98, 195
 Pullum Geoffrey, 37, 133, 136, 244
 Pustejovsky James, 148
 Putnam Hilary, 36, 38, 39, 206, 238
 Quine Willard, 1, 6, 31, 39, 40, 157,
 205–208, 235, 239, 245
 Rey Georges, 177, 180, 245
 Rooth Mats, 105, 190
 Rosenbloom Paul, 56, 239
 Ross John, 18, 76, 104
 Rosser Barkley, 1, 47, 55
 Russell Bertrand, 9, 46, 88, 166
 Sag Ivan, 23, 29, 59, 70, 235
 Sandra Dominiek, 40
 Santelmann Lynn, 169
 Scholz Barbara, 37, 133, 136, 244
 Schönfinkel Moses, viii, x, 1–4, 6, 7,
 25, 31–33, 44, 47, 51, 58,
 61, 76, 80, 109, 110, 113,
 200, 205–208, 223, 235,
 237, 239, 241, 253, 266
 Searle John, 177–180
 Shan Chung-Chieh, 195, 247
 Shaumyan Sebastian, 73, 101, 206,
 242
 Shi R., 169
 Shieber Stuart, 40, 70, 73–75, 241
 Simpson George, 247
 Siskind Jeffrey, 151, 157–159, 236,
 245
 Skolem Thoralf, 1, 91, 92
 Slobin Dan, 212
 Smith A., 149
 Smullyan Raymond, 45, 52, 56, 200,
 219, 239, 240, 247, 248
 Stabler Edward, 38
 Steedman Mark, viii, 4, 21, 24, 28,
 31, 54, 58, 64, 67, 71, 73,
 74, 78, 87, 89–91, 93, 97,
 98, 103, 106, 107, 110, 113,
 116, 119, 120, 125, 127,
 138–141, 148, 150, 152,
 153, 157, 160, 171, 184–
 189, 191, 201–203, 205,
 209, 211, 213, 238–241,
 243–247
 Steele Susan, 82, 173, 246
 Steels Luc, 106
 Stokhof M., 33
 Stoy J., 43
 Stromswold Karin, 212
 Szabolcsi Anna, 56, 64, 77, 78, 80–
 82, 87, 89, 92, 93, 101, 110,
 119, 163, 238–241
 Tardif Twila, 150
 Tarski Alfred, 88
 Tesnière Lucien, 16
 Thompson Sandra, 109
 Trechsel Frank, 119, 185
 Turing Alan, 22, 31, 36, 38–40, 103,
 136, 178, 213, 223, 225,
 237, 247
 Turner David, 56, 58, 239
 Ullman Jeffrey, 226
 van Hout Angeliek, 144, 146, 245
 Vijay-Shanker K., 73, 82, 83, 116,
 202
 Villavicencio Aline, 157, 191, 202
 Wackernagel Jacob, 63
 Wadler Philip, 195, 246
 Weinberg Amy, 225
 Weir David, 73, 82, 83, 116, 202
 Williams Edwin, 157, 245
 Wittgenstein Ludwig, 88, 135, 157,
 207, 244
 Yang Charles, 211, 213
 Yngve Victor, 206
 Zaenen Annie, 29, 142, 143

Zettlemoyer Luke, 150, 152, 153,
157, 160, 181, 191
Zurif Edgar, 41

Subject index

- accusative case, 170, 176
adjacency, viii, x, 6, 7, 205
agreement, 10, 11, 19, 70, 88, 110,
129, 165–167, 169–173,
230, 246
Aktionsart, 144–146, 148
Albanian, 104, 173
algorithm, 5, 16, 73, 149, 182, 199,
203, 219, 225, **226**, 246
applicative, system, 48, 68, 138, 183,
184, 206, 211, 223, 230,
239
Arabic, 12
argument sharing, 53
argumenthood, 34, 104, 125, 171,
193, 243
arity, Schönfinkel-Curry, **32**, 44, 109
autosegment, 164, 207

Bayesian, 152, 154
boundedness, 31, 121–124, 127, 128,
130, 131, 244

c-command, 88, 91, 92, 234, 242,
248
case marking, 173, 175, 230, 246
categorical grammar, ix, 10, 13, 65,
66, 73, 207
category, 28, 54, 65, 71, 129, **231**
distributional, 9, 72, 230
dollar, 71
exponent, 71, 95, 99, 102
formal, 66
functional, 110, 163–167, 169,
170
substantive, 66
type-dependence, 129
causative, 12, 13, 244
codeterminism, vii, x, 22, 59, 121

combinator, 1, 3, **219**
A, problem of, 190
B, 4, 16, 25, 32, 33, 44, 45, **49**,
50, 51, 76, 78–81, 87, 92,
95, 101, 109, 116, 187,
191, 192, 196, 201, 205,
209, **219**, 241
B², 4, 55, 58
C, 4, 45, **51**, 55, 62–64, 79–82,
93, **219**
D, 56, **57**, 239
fixpoint, 220
I, 32, **45**, 113, 134, 199, 223,
237
J, 45, **55**, 113
juxtaposition, 1
K, 36, 45, **48**, 49, 51, 79–81,
219, 221, 237, 242
O, 4, 5, 34, **57**, 113, 114, 116,
239
Φ, 45, 52, **53**, 54, 113
Ψ, 45, **54**, 55
power of, 32, 220
regular, 192, 201
S, 4, 22, 45, 51, **52**, 56, 77–80,
187, 197, 209, 219, 241,
247
S'', **58**, 82, 83, 117, 193
T, 5, 45, **47**, 58, 82, 83, 193,
194, 209, **219**
U, 37, 220, **247**
W, 3, 45, **49**, 80, **219**
X, 239
Y, 34, 35, 37, 45, **46**, 47, 55, 57,
133, 199, 209, 210, 220,
244
Z, **56**, 87, 92, 95–97, 100
complexity
computational, 149, 155,

- 225–228
 Kolmogorov-Chaitin, 241
 computation, tractable, **226**, 227
 computationalism, 37, 40, 73, 149, 220
 configuration, 225
 constituent, vii, x, 3, 5, 16, **22**, 138
 CCG, 61, 87, 89, 138, 141, 205, 208
 complete, 78, 79, 101
 impossible, viii, 13, 17, 101, 116, 188, 201, 209
 interpretable, vii, viii, 2, 9, 17, 43, 138, 208
 possible, viii, 3, 4, 7, 16, 21–24, 41, 48, 76, 79, 80, 83, 84, 138, 142, 168, 208, 241
 test, 23, 24, 26, 28, 29, 102, 236, 240
 constraint, 240
 computing, 22
 extraneous, 20, 106, 118, 120, 135, 152, 169
 formal, 89
 LEX, 93, 94, 125, 127, 128, 131, 147, 184, 185, 201
 local, 50, 76, 183
 multiple, 23, 60, 103, 129, 132
 nonlocal, 21
 semantic, 138, 162, 236
 substantive, 36, 38, 89, 103, 107, 110, 125, 182
 syntactic, 12, 18, 46, 71, 74, 76, 79, 102, 129, 147, 148, 183, 236, 242
 transderivational, 21
 type, 21
 universal, 104, 107, 110, 120, 130, 156, 184, 200
 Construction Grammar, 124, 143, 148, 208, 240
 control, 128
 Coordinate Structure Constraint (CSC), 18, 20, 21, 102, 236
 coordination, 17, 21, 22, 53, 63, 76, 78, 83, 84, 93, 100, 102, 113, 115, 119, 142, 236, 240, 245
 Curryng, 2, 3, 31, 44, 46, 58, 61, 68, 74, 104, 113, 118
 decidability, 40, 226
 dependency, 3, **233**
 crossing, 73–75, 241
 semantic, x, 1, 3, 4, 234
 syntactic, vii, 12, 40, 43, 45, 58, 199
 Dependency grammar, ix
 Deterministic Turing Machine, 226
 Distributed Morphology, 207, 236
 Dutch, 73, 74, 114, 118, 142–144, 147, 148, 245
 empty category, 2, 29, 40, 41, 94, 156, 210, 211
 English, 5, 6, 10, 11, 26, 39, 55, 79, 104, 107, 115, 116, 125, 127, 129, 130, 132, 138, 140, 166, 168, 169, 172, 188, 189, 209, 212, 230, 241, 242
 ergativity, 74, 76, 119, 129, 132, 170, 171, 212
 freedom, degrees of, 13, 16, 29, 40, 48, 63, 81, 102, 103, 122, 124, 128, 130, 148, 190, 206, 243
 function, principal, 61
 fusion, 51, 241
 gapping, 54, 66, 119, 186–188, 208, 239
 Generalized Phrase-structure Grammar (GPSG), 132, 211

- German, 81, 114, 118, 169
 Greek, 104
 Gusii, 63
 Göttingen, 1, 235

 head, 20, 21, 23, 35, 74, 76, 81, 100,
 113, 115, 124, 131, 145,
 147, 148, 164–167, 186,
 190, 194, 199, 203, 243
 Head-Categorial Uniqueness (HCU),
 110
 Head-driven Phrase-structure
 Grammar (HPSG), ix, 23,
 29, 235
 Hume
 question, ix, 16, 135, 209

 interdefinability, 48, 61, 80, 82, 83,
 240
 interpretable, immediately, viii, 2, 17
 intonation, 3, 22, 23, 60, 119, 134,
 138–141, 174, 175, 184

 juxtaposition, 7, 25, 27, 29, 31, 32,
 44, 47, 48, 50, 58, 61, 197,
 200, 202, 205, 217, 233,
 234

 K^wak^wala, 13, 14

 lambda calculus
 binding, 217
 conventions, 217
 conversion, **217**
 alpha (α), **217**
 beta (β), **217**, 218, 221
 eta (η), 6, **217**
 equivalence, 218
 normalization, **218**
 term, **217**
 Lexical Functional Grammar (LFG),
 23, 29, 149
 LF-command, 90
 linear-indexed grammar, 73, 229

 locality, domain of, 72, 99, 171, 243
 logical form (LF), 64, 85, 87–95, 98,
 100, 101, 103–105, 109,
 120, 142, 203, 205, 213,
 243

 merge, 24, 25, 58, 206, 211
 Mildly Context-sensitive Language
 (MCSL), 73
 minimalism, 24, 38, 179, 206
 monad, 95, 98, 102, 182, 183, 192,
 194–203, 205, 209, 246,
 247
 morphology, 11–13, 26, 27, 81, 123,
 125, 127, 131, 132, 147,
 156, 162, 169, 173, 174,
 179, 183, 189, 190, 207,
 235, 236, 244, 245
 movement, ix, 24, 25, 58, 97, 134,
 156, 206, 207, 211, 237

 normal-order evaluation, 218, 219

 order, of a function, 227
 Orifice, 116, 117
 OSV, 212, 242
 OVS, 212

 passive, 12, 13, 31, 99, 110, 119,
 121–128, 130, 142–145,
 147, 152, 243–245
 phonology, 3, 11–13, 23, 63, 140,
 175, 183, 191, 199, 203,
 235, 245
 predicate sharing, 55
 Predicate-argument Dependency
 Structure (PADS), 9, 11,
 35, 72, 87, 88, 104–106,
 120, 122, 125, 127–129,
 138, 140, 148, 151, 152,
 154, 156, 171–174, 176,
 182, 191, 202, **233**, 243,
 245

- primitive recursive function, 226
- Principle of Categorial Type
 - Transparency (PCTT), 107–110, 131, 148
- Principle of Lexical Head
 - Government (PLHG), 110, 148
- procedure, 226
- pronoun, 87–89, 91, 92, 94–100, 102–104, 108, 115, 122, 126, 242, 243
 - as argument, 95
 - as variable, 95
 - resumptive, 71, 102
- Radical Lexicalism, **66**, 72, 206
- radical lexicalization, x, 4, 28, 53, 59, 61, 70, 72, 73, 83, 101, 119, 124, 125, 128, 130, 135, 137, 143, 146, 148, 156, 157, 169, 172, 174, 175, 183, 184, 197, 206, 209, 211, **229**
- raising verb, 171
- recursion
 - semantic, 34, **37**, 133, 136, 209, 210, 221
 - syntactic, 37, **46**, 133, 134, 136, 209, **210**, 221
- reflexive, 31, 64, 87–89, 92–94, 103, 104, 121, 122, 127, 148, 172, 184, 243, 244
- reflexivization, 90, 91, 128
- relative pronoun, 70, 103, 129, 132
- relativization, 20, 21, 29, 35, 71, 74, 76, 84, 88, 100, 101, 119, 121, 128–132, 170
- resource, viii, 49, 74, 76, 119
 - computational, 40, 88, 149
 - grammatical, 173–175
 - insensitivity, 49
 - lexical, 10, 11, 194
 - morphological, 246
- right-node raising, 83, 84, 93
- rule-to-rule hypothesis, 59, 72, 110, 121, 130, 131, 205, 206
- semi-decides, 226
- Separation Hypothesis, 12, 236, 245
- sequencer, 192
- seriation, 210, 213
- SKIM, 235
- slash
 - modality, 68
 - underspecification (‘|’), 70, 198
- SOV, 189, 212
- structure dependence, viii, 17–19, 21, 22, 24, 28, 63, 98, 166, 175, 210, 211, 237, 240
- subcombinator, **35**
- subordinator, 163
- supercombinator, 33–36, 57, 58, 114, **220**, 221
- SVO, 67, 166, 188, 212
- Swiss German, 73–75
- syntacticization, vii, viii, 4–6, 9, 10, 13, 16, 21, 25, 34, 41, 43–59, 61, 62, 66–70, 74, 76–78, 100, 114, 118, 121, 130, 136, 138, 164, 183, 190, 193, 198, 199, 205, 207–210, 221, 238, 239
- syntactocentrism, x, 72
- telicity, 142–148, 245
- topic prominence, 119
- topicalization, 29, 121, 123
- transformation, 18, 19, 21, 23, 24, 28, 29, 36–38, 88, 89, 91, 94, 110, 124, 163, 206, 207, 210, 233, 236, 237, 240
- Turing Machine, 225
- Turing representability, 36, 40, 72, 225, **226**
- Turkish, 20, 21, 35, 63, 64, 66, 71, 83, 84, 99, 105, 115, 118,

- 119, 123–125, 127–129,
140, 157, 169, 174–176,
186, 189, 212, 241, 244,
246, 247
- type, **9**, 165
 combinatory, 10, **230**
 distributional, 230
 semantic, **231**
 subtyping, ix
 syntactic, 10, 65
- type dependence, viii, 11, 17–22, 28,
61, 98, 99, 124, 130, 237
- type raising, 5, 48, 64, 83, 84, 87, 91,
92, 94, 102–104, 124, 125,
127, 163, 165–168, 170,
172, 173, 175, 188, 191,
193, 194, 200, 242
- Type-logical grammar, ix, 29, 65
- Tzeltal, 155, 156
- unaccusative, 144, 147, 185
- unboundedness, 35, 121, 122,
128–132
- undecidable, 226
- unergative, 142, 144, 147, 185
- unification, ix, 70, 120
- universal grammar, 74, 82, 135, 153,
156, 181, 203, 210, 211,
213
- vacuous abstraction, 33, 80, 81
- value-raising, 163, 166, 171, 172
- variable, 3, 87
 bound, **217**
 free, **217**
- variable-free semantics, 100–103
- VOS, 82
- VSO, 10, 64, 79, 81, 82, 93, 94, 97,
125, 127, 166, 188, 197,
212, 242
- Washo, 124
- Welsh, 10, 64, 79–82, 94, 97, 99,
125–127, 147, 166, 170,
189, 197, 242
- word order, 64, 66, 76, 79, 82, 83, 88,
119, 129, 132, 156, 169,
170, 173–175, 197, 212,
237, 244, 246
- wrap, ix, 29, 61–64, 82, 87, 93, 94,
97, 102, 120, 205, 242

