Cross-lingual Semantic Parsing with Categorial Grammars

Kilian Evang

Cross-lingual Semantic Parsing with Categorial Grammars

Kilian Evang

The work in this thesis has been carried out under the auspices of the Center for Language and Cognition Groningen of the Faculty of Arts of the University of Groningen.

Groningen Dissertations in Linguistics 155
ISSN: 0928-0030
ISBN: 978-90-367-9474-9 (printed version)
ISBN: 978-90-367-9473-2 (electronic version)

Document prepared with LaTeX 2ε and typeset by pdfTeX (TeX Gyre Pagella font)
Cover design by Kilian Evang
Printed by CPI Thesis (`www.cpithesis.nl`) on Matt MC 115g paper

rijksuniversiteit
groningen

Cross-lingual Semantic Parsing with Categorial Grammars

Proefschrift

ter verkrijging van de graad van doctor aan de
Rijksuniversiteit Groningen
op gezag van de
rector magnificus prof. dr. E. Sterken
en volgens besluit van het College voor Promoties.

De openbare verdediging zal plaatsvinden op

donderdag 26 januari 2017 om 12.45 uur

door

Kilian Evang

geboren op 1 mei 1986
te Siegburg, Duitsland

Promotores
Prof. dr. J. Bos

Beoordelingscommissie
Prof. dr. M. Butt
Prof. dr. J. Hoeksema
Prof. dr. M. Steedman

Acknowledgments

Whew, that was quite a ride! People had warned me that getting a PhD is not for the faint of heart, and they were right. But it's finally done! Everybody whose guidance, support and friendship I have been able to count on during these five years certainly deserves a big "thank you".

Dear Johan, you created the unique research program on deep semantic annotation that lured me to Groningen, you hired me and agreed to be my PhD supervisor. Moreover, you gave me lots of freedom to develop and try out my ideas, always listened to me, always believed in me and always, always had my back in easy times and in difficult times. Thank you so much!

Besides my supervisor, a number of other people have also helped with shaping and focusing my research. I would especially like to thank two pre-eminent experts on CCG parsing, Stephen Clark and James Curran. Discussing things with them helped me a great deal with getting my thoughts in order, inspired new ideas and encouraged me to pursue them. I would also like to thank Laura Kallmeyer and Frank Richter for their invaluable mentorship during the time before my PhD studies.

Dear professors Miriam Butt, Jack Hoeksema and Mark Steedman—thank you for agreeing to be on the reading committee for my thesis! I am honored that you deem it scientifically sound. I apologize for not having had the time to address all of your—justified—suggestions for further improvement. I will heed them in my future research.

Few things are as important to a researcher as a good working environment, and I have been especially fortunate here. I would like to start by thanking my fellow PhD students Johannes, Dieke, Rob, Hessel and

Rik, as well as Vivian and Yiping. We make a fine pubquiz team even though we suck at recognizing bad music from the 70's. Thank you also for the coffee at Simon's, the beer at Hugo's and more generally for keeping me sane and happy during the final year of my PhD. I will try to return the favor when it comes your turn. Dear Dieke and Rob, special thanks to you for agreeing to stand by my side as *paranimfen* on the day of my defense!

Next, I would like to thank the whole Alfa-informatica department, that entity that formally does not exist, but is united by computational linguistics, the Information Science degree program, the reading group, the corridor and the greatest collegial spirit one could wish for. Dear Antonio, Barbara, Dieke, Duy, George, Gertjan, Gosse, Gregory, Hessel, Johan, John, Lasha, Leonie, Malvina, Martijn, Pierre, Rik and Rob—thank you for being great colleagues and coworkers in all matters of research, teaching and beyond. Congratulations for upholding that spirit for 30 years—and here's to the next 30!

Within the entire Faculty of Arts and its various organizational units, I would especially like to thank Alice, Karin, Marijke and Wyke for being incredibly helpful and friendly with all things related to training, teaching and administration throughout my 5+ years here.

Although I cannot name all former coworkers here, I would like to thank some of them, and their families, specially. Dear Valerio and Sara, thanks for being like a wise big brother and awesome sister-in-law to me. Dear Noortje and Harm, thank you for never ceasing to infect me with your energy and your quick and big thinking. Dear Elly, thank you for being a great help and wonderful friend. Dear Gideon, thanks for being such a cool buddy.

Thanks are due also to my friends outside work, the wonderful people I met at Gopher, the USVA acting class, the "Kookclub" and elsewhere, who have been making Groningen feel like home. I would especially like to thank Anique, Aswin, Evie, Hylke, Maaike, Marijn, Ni, Qing, Rachel, Rogier and Yining. Thank you for all the great times, here's to many more!

Furthermore, I am extremely grateful for all friendships that have stood the test of time and geographical distance—not in all cases evidenced by frequent contact, but by the quality of contact when it hap-

pens. Dear Malik, Regina, Anne, Johannes, Anna, Katya, Norbert, Aleks, Anna, Laura, Marc, Chris, Laura, Nadja, Nomi, Chrissi, Armin and Martini; dear members of the *Gesellschaft zur Stärkung der Verben*; my dear Twitter timeline—thank you for your friendship, for our inspiring exchanges and for our common projects. The author of this thesis would not be the same person without you.

To an even greater extent, this holds, of course, for my family. Dear Mom and Dad, thank you for giving me all the freedom, support, stability and love one could wish for, for 30 years and counting. Dear Viola and Valentin, thank you for being fantastic siblings! Dear *Oma*, Roselies, Markus, Peter, Heike, Karin, Dany, Bille, Brigitte, Matze, Tina and the whole lot—you are a great family. Thank you for always making me feel good about where I come from, wherever I may go next.

Groningen, December 2016

Contents

Chapter 1

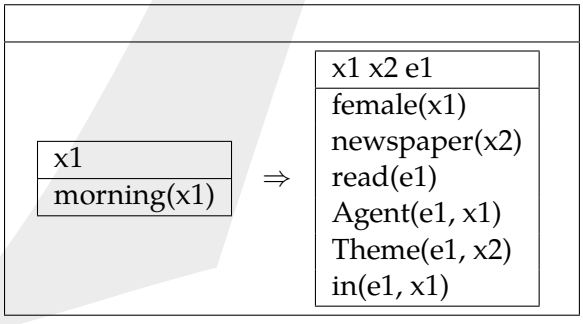
Introduction

Imagine you are trying to learn an unknown language—say, Dutch. All you have is a list of Dutch sentences with their English translations. For example:

(1) a. Zij leest elke morgen de krant.
b. She reads the newspaper every morning.

Luckily, you can understand English. That is, when you read an English sentence, you form a semantic interpretation of it in your mind: you know what the sentence means. So, if you trust the translation, you already know what *Zij leest elke morgen de krant* means as well. If you are a formal semanticist, you might diagram the meaning as follows:

(2)



But in order to learn the language, you have to find out what *individual words* mean and how they combine. Only then will you be able to recognize and interpret them when reading Dutch in the future, gauging the meaning of texts you are not given translations for.

But which word means what? You could start with the assumption that Dutch is fairly similar to English, and hypothesize that the words correspond to each other one-to-one: *Zij=She*, *leest=reads*, *elke=the*, *morgen=newspaper*, *de=every*, *krant=morning*.

However, as you read more sentences, you notice that *de* is an extremely frequent word. It is often found in sentences whose English translations do *not* contain *every*. But it tends to co-occur with *the*. So it seems more likely that *de* is in fact the definitive article.

Thus, you change your hypothesis, and now assume the Dutch sentence has a different order from the English one, *elke* meaning *every*, *morgen* meaning *morning*, and *krant* meaning *newspaper*. That seems better, also because the mapped words now sound more similar to each other, a strong clue in the case of related languages. You can corroborate your hypothesis by verifying that *krant* is frequently found in other sentences whose translation contains *newspaper* or a semantically equivalent word such as *paper*.

As you study this sentence and others, you will also notice that the changed word order is a frequent pattern: an adverbial intervenes between verb and object. The more sentences you study and try to understand, the more words you learn, and the more easily you can follow Dutch word order.

This is a process by which a dedicated human is conceivably able to learn to read a language. Can computers learn a language in a similar way? Let us start with the assumption that a computer already understands English. This assumption is not as presumptuous today as it was ten years ago. Great progress has been made in the young field of *semantic parsing*, the mapping of natural-language sentences to formal meaning representations such as the one in the diagram above. Once a computer has mapped natural-language input to such a formal representation, it can use it to draw conclusions and take the appropriate actions—think of a robot servant that follows your instructions, or of a program that reads medical papers and finds new cures by using in-

formation previously reported, but not combined by human scientists. And that is what we mean by computers "understanding": taking appropriate actions and producing new insights.

Despite recent successes, semantic parsing remains a very difficult task, especially when broad coverage of linguistic domains like newswire text or everyday speech is required. Existing systems typically parse only one language, and that is typically English: computers are monolinguals. Countless person hours have gone into engineering computational grammars and semantically annotating texts to teach computers that one language. We should not start from scratch for other languages.

Instead, one possibility is to use *cross-lingual learning*, a family of methods that train natural-language processing systems with few or no manually created resources specifically for the target language. Cross-lingual learning draws on resources for another language and on parallel corpora (collections of translated text) to transfer knowledge from source-language systems to target-language systems—much like the learning process sketched above. Seeing the considerable cost of manually creating resources for semantic parsers, computers that learn to "understand" language in this way would be very useful.

1.1 About this Thesis

This thesis deals with the problem of learning a broad-coverage parser cross-lingually. We are particularly interested in finding a method that assumes little to no available resources (such as grammars or lexicons) for the target language, so that it is applicable to under-resourced languages. The problem is addressed within the framework of Combinatory Categorial Grammar (CCG) as a grammar formalism and Discourse Representation Theory (DRT) as a meaning representation language. This leads to the following four research questions:

(i) Does CCG have the flexibility required for applying it to diverse natural languages, meaning representation formalisms and parsing strategies?

(ii) Broad-coverage semantic parsing requires training data in the form of text annotated with suitable meaning representations such as Discourse Representations Structures (DRS). How can the knowledge of humans be used effectively for building such a corpus?

(iii) One type of cross-lingual learning is *annotation projection*, the projection of source-language annotations to target-language annotations, followed by training on the target-language data so annotated. In the case of CCG derivations, annotation projection amounts to automatic parallel semantic treebanking:

- Is there an algorithm for doing this?
- Can it deal with translation divergences?
- Does it produce linguistically adequate analyses?

(iv) How can such projected derivations be used to train a broad-coverage semantic parser?

The thesis is structured as follows.

Part I provides the background: Chapter 2 introduces the necessary background on CCG, and Chapter 3 reviews existing approaches to semantic parsing.

Part II deals with narrow-coverage semantic parsing, i.e., semantic parsing of utterances geared towards specific, concrete tasks to be performed by the understanding system. We address question (i) in this setting by applying CCG to a new semantic parsing task dealing with situated robot commands. Despite being narrow-coverage, the task is challenging and tests CCG's flexibility in several ways, including unedited language, non-standard meaning representations and interfacing with a spatial planner.

Part III deals with broad-coverage semantic parsing, i.e., semantic parsing of text not necessarily geared to any specific task. Chapter 5 addresses question (ii) by describing and evaluating the annotation scheme and methodology used in the Groningen Meaning Bank project. Chapter 6 addresses question (iii) by describing and evaluating an algorithm for projecting CCG derivations across parallel corpora. Chapter 7 addresses question (iv), developing a cross-lingual semantic parser learner

based on the projection algorithm and evaluating it on an English-Dutch parallel dataset.

Part IV concludes with Chapter 8, formulating the answers to the research questions.

1.2 Publications

Some chapters of this thesis contain material from or are extended versions of peer-reviewed publications:

Chapter 4 is an extended version of Evang and Bos (2014).

Chapter 5 contains material from Basile, Bos, Evang and Venhuizen (2012) , Evang and Bos (2013) and Bos, Basile, Evang, Venhuizen and Bjerva (2017) .

Chapter 7 is an extended version of Evang and Bos (2016).

Part I

Background

Chapter 2

Combinatory Categorial Grammar

2.1 Introduction

In this thesis, we are concerned with automatically deriving the meanings of sentences. It is much easier to get a hold on this task if we assume that the meaning of a sentence follows via a small set of rules from the meanings of the parts it consists of (its *constituents* or *phrases*), and the meaning of each part in turn follows via these rules from the meanings of its subparts—ultimately, everything is composed of the meanings of individual *words*. This view is known as the *principle of compositionality* and, as Janssen (2012) argues, was first formulated by Carnap (1947). We adopt it in this thesis. We will call the formal meaning representations assigned to words, sentences and other constituents their *interpretations*.

For specifying the set of rules that govern how word interpretations are put together into constituent and sentence interpretations, our tool of choice is Combinatory Categorial Grammar (CCG; Steedman, 2001). CCG is unique in that it couples syntax and semantics very tightly. It keeps stipulations about the syntax to a minimum, notably having a very flexible notion of constituency. It is this flexibility, together with a clear, principled approach to deriving interpretations, and a proven

track record as a framework for building robust statistical parsers, that make CCG appear as a suitable framework for the cross-lingual parsing work tackled in this thesis. In this chapter, we introduce CCG with a focus on the aspects especially important for this thesis.

2.2 Interpretations

In CCG, the interpretations of words, sentences and other constituents are terms of the λ-calculus (Church, 1932; Barendregt, 1984, Chapter 2). The set of λ-terms over an infinite set V of variables is the least set Λ satisfying

1. $x \in \Lambda$ if $x \in V$,
2. $(\lambda x.M) \in \Lambda$ if $x \in V$ and $M \in \Lambda$ (λ-abstraction),
3. $(M@N) \in \Lambda$ if $M, N \in \Lambda$ (function application).

We define a *substitution* $\cdot[x := N]$ with $x \in V$, $N \in \Lambda$ as the least partial function from Λ to Λ satisfying

1. $x[x := N] = N$ if $x \in V$,
2. $(\lambda x.M)[x := N] = (\lambda x.M)$,
3. $(\lambda y.M)[x := N] = (\lambda y.M[x := N])$ if $x \neq y$ and y does not occur in N[1],
4. $(A@B)[x := N] = (A[x := N]@B[x := N])$.

This definition prevents accidental binding of variables in the substituted expression N by having substitution be undefined for such cases.

The equivalence relation $\equiv$ between terms in Λ is the least symmetric and transitive relation from Λ to Λ satisfying

[1] Substitution can be defined more leniently so that only *free* occurrences of y in N are disallowed, but this does not change the equivalence relation defined below because it allows for free renaming of variables.

1. $x \equiv x$ if $x \in V$,
2. $(\lambda x.M) \equiv (\lambda x.N)$ if $M \equiv N$,
3. $(M@N) \equiv (A@B)$ if $M \equiv A$ and $N \equiv B$,
4. $(\lambda x.M) \equiv (\lambda y.M[x := y])$ if y does not occur in M (α-conversion),
5. $((\lambda x.M)@N) \equiv M[x := N]$ (β-conversion).

For example, the λ-terms $(a@b)$, $((\lambda x.(x@b))@a)$ and $((\lambda y.(y@b))@a)$ are all equivalent. We will not usually distinguish between equivalent λ-terms, instead speaking about them as if they were the same.

We use the double-bracket notation to refer to the interpretation of a word, e.g., $[\![Mary]\!]$ is the interpretation of the word *Mary*. What those interpretations look like exactly depends on the semantic formalism one uses. For example, with a simple formalism based on predicate logic, we might have $[\![Mary]\!] = m$, $[\![John]\!] = j$ and $[\![loves]\!] = (\lambda x.(\lambda y.((love@y)@x)))$—or, written with some syntactic sugar: $[\![loves]\!] = \lambda x.\lambda y.love(y, x)$. As the interpretation of the sentence *John loves Mary*, we might wish to obtain $(([\![loves]\!]@[\![Mary]\!])@[\![John]\!]) = love(j, m)$.

To derive interpretations for sentences compositionally, we must restrict the set of possible constituents and their interpretations. CCG does this by assigning each word a *syntactic category* and providing a number of *combinatory rules* through which constituents may be derived.

2.3 Syntactic Categories

A syntactic category is either a *basic category* or a *functional category*. To a first approximation, the basic categories are:

- N: noun, e.g., *book*,
- NP: noun phrase, e.g., *a book* or *Mary*,
- PP: prepositional argument, e.g., *to John*,
- S: sentence, e.g., *Mary gave a book to John*.

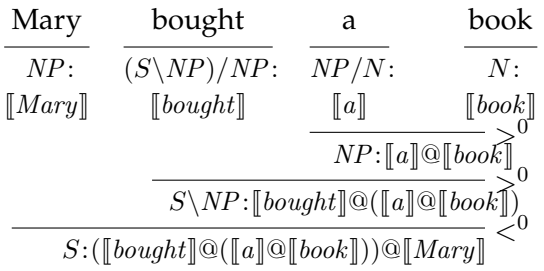


Figure 2.1: CCG derivation diagram of the sentence *Mary bought a book*.

Many well-known syntactic constituent types—such as *determiner*, *verb phrase* or *preposition*—do not have basic categories. Instead, they have *functional categories* which indicate their potential to combine with other constituents to form new constituents. Functional categories are of the form (X/Y) or $(X\backslash Y)$ where X and Y are syntactic categories. X is called the *result category* and Y is called the *argument category*. Intuitively, category (X/Y) on a constituent indicates that the constituent can combine with a constituent with category Y on its immediate right to form a new constituent with category X. The first constituent is then called the *functor*, the second the *argument* and the third the *result*. Similarly, category $(X\backslash Y)$ on a constituent (the functor) indicates that it can combine with a constituent with category Y on its immediate left (the argument) to form a new constituent with category X (the result).

For example, an intransitive verb such as *walks* can combine with a (subject) noun phrase such as *Mary* on its left to form a sentence such as *Mary walks*, so it has the category $(S\backslash NP)$. A transitive verb such as *bought* has the category $((S\backslash NP)/NP)$, saying that it can first combine with an (object) NP such as *a book* on its right to form a constituent with category $(S\backslash NP)$ such as *bought a book*—which can in turn combine with an NP on its left to form a sentence. In general, CCG verb categories are designed so that arguments combine in order of decreasing obliqueness. This is motivated by binding theory, for details see Steedman (2001, Section 4.3).

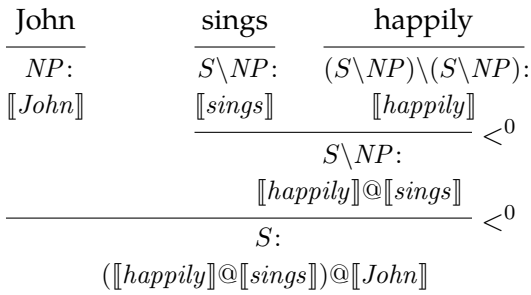


Figure 2.2: Derivation with a VP modifier.

A complete analysis—called a *derivation* in CCG—of a sentence is shown in Figure 2.1. Each constituent is drawn as a horizontal line with its category written underneath, followed by a colon, followed by its interpretation. Here and from now on, we drop outermost parentheses both on categories and on interpretations, to avoid notational clutter. *Lexical* constituents are written at the top, with the corresponding words above the line. *Non-lexical constituents* are drawn underneath their children, and on the right of the line a symbol is drawn indicating the combinatory rule (see below) that licenses the constituent, given its children.

We think of every constituent as having a list of *argument slots*, each associated with a category and a slash direction. For constituents with basic categories, this is the empty list. For constituents with category X/Y or $X\backslash Y$, it is a list headed by an argument slot associated with $/Y$ resp. $\backslash Y$, followed by the elements of the argument slot list a constituent with category X would have. When a constituent serves as functor in the application of a binary rule, we say that its first argument slot is *filled* in this process, i.e., it no longer appears on the result constituent. For example, a transitive verb (category $(S\backslash NP)/NP$) has two NP argument slots, the first for the object, the second for the subject. After combination with the object, only the subject argument slot remains on the result constituent (category $S\backslash NP$).

Constituents with categories of the form X/X or $X\backslash X$ are called *modifiers*. They modify the interpretation of a constituent but do not

change its category, i.e., the categories of argument and result are the same. Examples of modifiers are attributive adjectives, which modify nouns, and adverbs, which modify, e.g., verb phrases. An example of VP modification is given in Figure 2.2.

2.3.1 Morphosyntactic Features

We have so far presented a simplified set of basic categories: $\{S,\ NP,\ N,\ PP\}$. In actual grammars, often instead of the atomic basic category NP complex basic categories such as $NP[3s]$ are used, each with morphosyntactic features for person, number, gender and so on. This allows grammars to implement selectional restrictions such as the fact that a third-person singular verb requires a third-person singular subject. Not all NP categories need to have all features. A category with a missing feature is thought of as underspecified, i.e., as having a variable for this feature which can take on any value via unification when a combinatory rule is applied. For example, a third-person singular verb might have category $(S\backslash NP[3s])/NP$. The first argument category, corresponding to the direct object, is underspecified for person and number, so any NP will be able to serve as the object, regardless of its person and number (cf. Steedman, 2001, Section 3.1).

The NP features appear not to be essential for statistical parsing of English, and we do not use them in this thesis. However, we do distinguish clause types via a single S feature, following statistical parsing work of Hockenmaier (2003a) and Clark and Curran (2007). Relevant clause categories for English are:

- $S[\mathrm{dcl}]$: declarative sentence, e.g., *Mary wants to give John a book*,
- $S[\mathrm{wq}]$: wh-question, e.g., *What are you doing*,
- $S[\mathrm{q}]$: yes-no-question, e.g., *Are you crazy*,
- $S[\mathrm{qem}]$: embedded question, e.g., *how many troops have arrived*,
- $S[\mathrm{em}]$: embedded declarative, e.g., *that you are crazy*,
- $S[\mathrm{frg}]$: sentence fragment, e.g., *not exactly inevitable*,

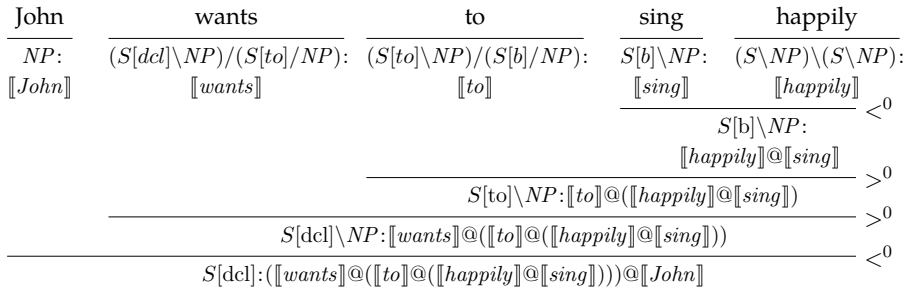


Figure 2.3: Derivation illustrating the use of category features: the categories of *wants* and *to* select for specific VP types whereas the modifier *happily* is underspecified.

- $S[\text{for}]$: small clause headed by *for*, e.g., *for one consultant to describe it as "clunky"*,
- $S[\text{intj}]$: interjection, e.g., *Wham!*,
- $S[\text{inv}]$: elliptical inversion, e.g., *may the Bush administration* in *so may the Bush administration*,
- $S[\text{b}]\backslash NP$: VP headed by bare infinitive, subjunctive or imperative, e.g., *give John a book*,
- $S[\text{to}]\backslash NP$: VP headed by *to*-infinitive, e.g., *to give John a book*,
- $S[\text{pss}]\backslash NP$: VP headed by passive past participle, e.g., *given a book by Mary*,
- $S[\text{pt}]\backslash NP$: VP headed by active past participle, e.g., *given a book to John*,
- $S[\text{ng}]\backslash NP$: VP headed by by present participle, e.g., *giving a book to John*,
- $S[\text{adj}]\backslash NP$: predicative adjective phrase, e.g., *hypnotically grotesque*.

Complementizers and auxiliary verbs as well as adjectives, adverbs and verbs with clausal complements select for clauses with specific features whereas sentence/VP modifiers are typically underspecified for clause type and can apply to any sentence or any VP. Examples of both selection and underspecified modification can be seen in Figure 2.3.

2.4 Combinatory Rules

How constituents can combine is spelled out by CCG's combinatory rules. A combinatory rule applies to one or two adjacent constituents with particular categories as input and produces an output constituent whose category and interpretation depend on those of the input constituents. Combinatory rules are stated in the form $\alpha : f \Rightarrow \gamma : h$ (unary rules) or $\alpha : f \ \beta : g \Rightarrow \gamma : h$ (binary rules) where α, β are categories of input constituents, f, g are interpretations of input constituents, γ is the category of the output constituent and h is its interpretation.

2.4.1 Application

Above, we gave the intuitive meaning of functional categories as being able to combine with adjacent arguments of the argument category to yield a result of the result category. The rules of *forward application* and *backward application* implement this meaning:

(1) *Forward application ($>^0$)*
$(X/Y) : f \quad Y : a \quad \Rightarrow \quad X : (f@a)$

(2) *Backward application ($<^0$)*
$Y : a \quad (X\backslash Y) : f \quad \Rightarrow \quad X : (f@a)$

Semantically, the application rules implement function application: if the functor has interpretation f and the argument has interpretation a, the constituent resulting from application has interpretation $(f@a)$.

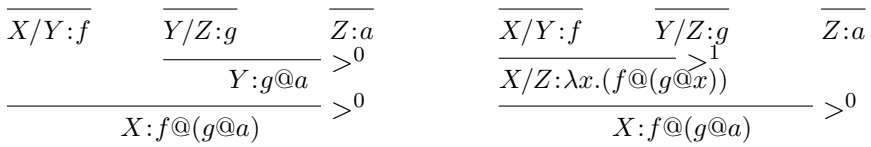


Figure 2.4: Two semantically equivalent derivations.

2.4.2 Composition

Consider, in the abstract, two adjacent constituents with categories X/Y and Y/Z. We can tell from the categories that if we have another constituent with category Z to the right of them, we will be able to derive a constituent with category X, as shown on the left in Figure 2.4. We could come up with a category that encodes this prediction: look for Z on the right, the result is X, i.e., X/Z. We may feel entitled to assign this category to a constituent spanning the constituents with categories X/Y and Y/Z—if we make sure the interpretation we eventually derive will be the same. This, too, is easy: the Y/Z constituent is waiting for a Z constituent with some interpretation—let's call it x—and when applied to it would yield the interpretation $g@x$, which would then be the argument to f. We can anticipate this process by using λ-abstraction, abstracting over the interpretation x that we will combine with later. The interpretation for our X/Z constituent is thus: $\lambda x.f@(g@x)$.

In fact, CCG has a combinatory rule that allows us to derive this "non-canonical" constituent, called (harmonic) *forward composition* ($>^1$). We say that the filling of the Z argument slot has been *delayed*. It can now be filled by combining the X/Z constituent with the Z constituent via forward application. We obtain a constituent with category X and interpretation $(\lambda x.(f@(g@z)))@a$, which is equivalent to $f@(g@a)$. Thus, we obtained the exact same constituent as before via a different derivation.

This may seem redundant and pointless until one realizes that some linguistically motivated analyses are only possible when we permit ourselves to derive such "non-canonical" constituents, for they can be ex-

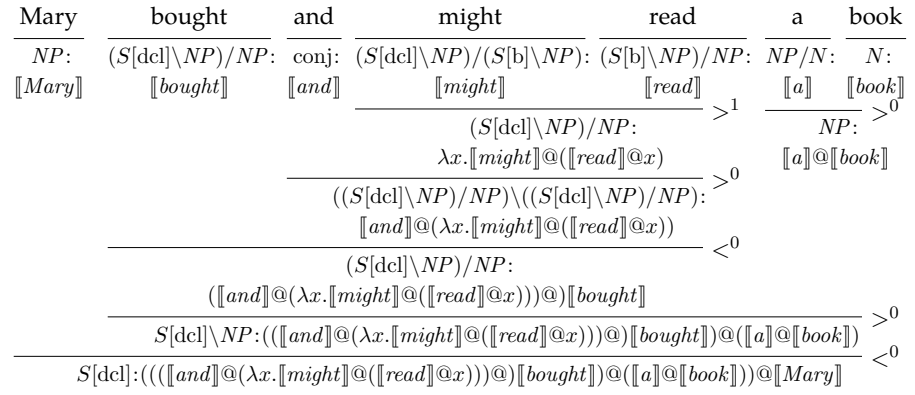


Figure 2.5: Example of non-constituent coordination. We abbreviate $(((S[\text{dcl}]\backslash NP)/NP)\backslash((S[\text{dcl}]\backslash NP)/NP))/((S[\text{dcl}]\backslash NP)/NP)$ as conj here.

plicitly required as arguments by other constituents such as conjunctions or relative pronouns. One example is the coordination of non-canonical constituents as analyzed in Figure 2.5. *read* is a transitive verb that expects an object on its right, as is *bought*. *might*, however, is a VP modifier that expects a constituent of category $S\backslash NP$ as argument, with no open object argument slot. Yet *might read* is a conjunct in the coordination, *a book* being the object to both *bought* and *read*. Composition allows for the correct analysis. CCG's harmonic composition rules are:

(3) *Forward harmonic composition ($>^1$)*
$(X/Y):f \quad (Y/Z):g \quad \Rightarrow \quad (X/Z):(\lambda x.(f@(g@x)))$

(4) *Backward harmonic composition ($<^1$)*
$(Y\backslash Z):g \quad (X\backslash Y):f \quad \Rightarrow \quad (X\backslash Z):(\lambda x.(f@(g@x)))$

Pairs of derivations such as those in Figure 2.4, where the categories of the words and of the largest constituents are identical and where the latter *necessarily* has the same interpretation even if the interpretations of individual words are not known, are called *semantically equivalent* (Eisner, 1996). Practically every real-world CCG derivation has multiple different semantically equivalent but distinct derivations, a characteris-

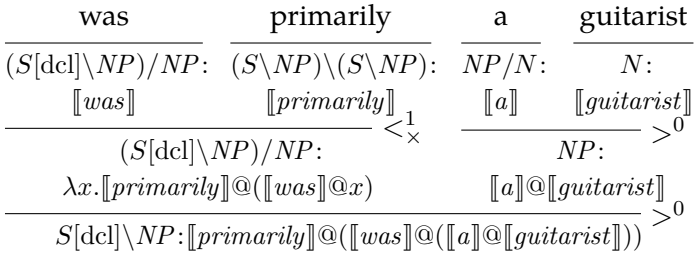


Figure 2.6: Internal VP modification analyzed using crossed composition.

tic of CCG that is known as *spurious ambiguity*.

2.4.3 Crossed Composition

Occasionally we would like to derive a constituent with interpretation $f@(g@a)$ but cannot do so with application and harmonic composition alone because the constituents with interpretations g and a are not adjacent; they are separated by the one with f. This is the case for example when a modifier appears inside the hypothetical constituent that it modifies, such as the modifier *primarily* in Figure 2.6—it is a VP modifier appearing between the verb and the argument it needs to make the VP complete.

This problem is solved by *crossed composition*. It is like harmonic composition, except that the slashes associated with the first argument slots of the input categories lean different ways:

(5) *Forward crossed composition ($>^1_\times$)*
$(X/Y){:}f \quad (Y\backslash Z){:}g \quad \Rightarrow \quad (X\backslash Z){:}(\lambda x.(f@(g@x)))$

(6) *Backward crossed composition ($<^1_\times$)*
$(Y/Z){:}g \quad (X\backslash Y){:}f \quad \Rightarrow \quad (X/Z){:}(\lambda x.(f@(g@x)))$

This way, the modifier can combine with the verb, delaying the application to the object. The resulting constituent is then adjacent to the object and can combine with it.

2.4.4 Generalized Composition

CCG's *generalized composition* rules allow for delaying not just one, but a number n of arguments. Application and composition rules are then special cases of generalized composition where $n = 0$ and $n = 1$, respectively. Following the notation of Vijay-Shanker and Weir (1994), we define generalized composition as follows:

(7) *Generalized forward composition ($>^n$)*
$(X/Y){:}f \quad (\cdots(Y|_1Z_1)|_2\cdots|_nZ_n){:}g \quad \Rightarrow$
$(\cdots(X|_1Z_1)|_2\cdots|_nZ_n){:}$
$\lambda x_1.\lambda x_2.\cdots\lambda x_n.(f@\cdots(((g@x_1)@x_2\cdots)@x_n))$
where $n \geq 0$, $X, Y, Z_1, ..., Z_n$ are categories, $|_1, ..., |_n \in \{/, \backslash\}$.

(8) *Generalized backward composition ($<^n$)*
$(\cdots(Y|_1Z_1)|_2\cdots|_nZ_n){:}g \quad (X\backslash Y){:}f \quad \Rightarrow$
$(\cdots(X|_1Z_1)|_2\cdots|_nZ_n){:}$
$\lambda x_1.\lambda x_2.\cdots\lambda x_n.(f@\cdots(((g@x_1)@x_2\cdots)@x_n))$
where $n \geq 0$, $X, Y, Z_1, ..., Z_n$ are categories, $|_1, ..., |_n \in \{/, \backslash\}$.

Some versions of CCG require $|_1, ..., |_n$ to all have the same direction (cf. Kuhlmann et al., 2015), but we do not require this. Forward composition with $n \geq 1$ is called *harmonic* if $|_1 = /$ and *crossed* if $|_1 = \backslash$, and vice versa for backward composition. We will mark crossed rule instances with a $\times$ subscript, e.g., $>^1_\times$ and $<^1_\times$. In practice, there is an upper bound on n depending on properties of the language. For English it is usually set as $n \leq 3$.

2.4.5 Type Raising

Sometimes a functor needs to combine with an argument but still has extra argument slots beyond the one corresponding to that argument. This is the case for example in object relative clauses, as in *the book that Mary bought*. Here, there is no object NP on the right of the verb, yet we wish to combine the transitive verb with category $(S\backslash NP)/NP$ with its subject NP to the left. The situation is opposite to the one that composition deals with: now, not the argument, but the functor, has an extra open argument slot. CCG deals with this by using composition, but

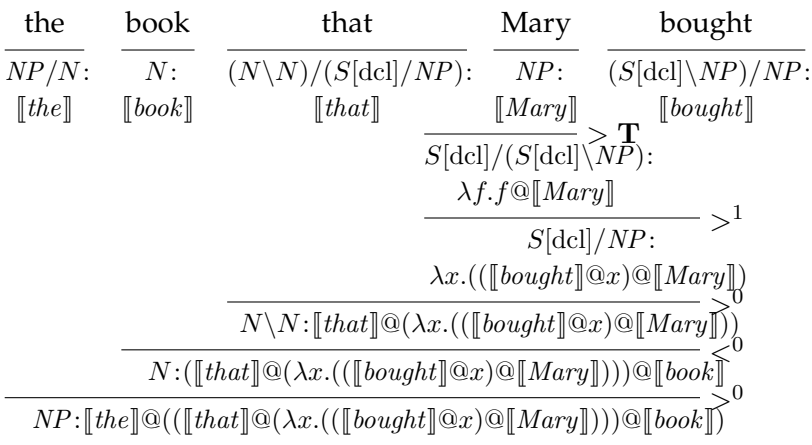


Figure 2.7: Analysis of a relative clause using type raising and composition.

first turning the argument into the functor and thereby the functor into the argument. This is done by the unary rules *forward type raising* ($>$ **T**) and *backward type raising* ($<$ **T**). An example derivation is shown in Figure 2.7.

Type raising can be motivated as follows: assume a constituent labeled $X:a$. Now if there is a constituent labeled $(T\backslash X):f$ on its right, they can combine via backward application into a constituent labeled $T:(f@a)$. If we relabel the first constituent $(T/(T\backslash X)):\lambda x.(x@a)$, we can derive the exact same constituent, only this time the first constituent is the functor and the second the argument, and we use forward instead of backward application. A similar argument can be made for the mirror-image case. Thus, type raising is defined as follows:

(9) *Forward type raising ($>$ **T**)*
$Y : g \Rightarrow (T/(T\backslash Y)):\lambda f.(f@g)$
where Y, T are categories.

(10) *Backward type raising ($<$ **T**)*
$Y : g \Rightarrow (T\backslash(T/Y)):\lambda f.(f@g)$

where Y, T are categories.

2.4.6 Rule Restrictions

If all existing composition and type-raising rules could be applied without limit, grammars would overgenerate and permit ungrammatical word orders. One solution to this problem is stipulating language-specific ad-hoc rule restrictions, stating that certain rules can only apply to certain input categories, or not at all. For example, Steedman (2001) stipulates that in English, forward crossed composition is banned and backward crossed composition can only apply when the second input is a clause modifier but not, for example, when it is a noun modifier. For Dutch, he subjects composition rules to even more complex restrictions making reference, e.g., to a syntactic features that distinguish main clauses from subordinate clauses.

Baldridge (2002); Baldridge and Kruijff (2003) propose the *multimodal* extension to CCG where language-specific restrictions are placed in the lexicon rather than in the rules, resulting in truly universal, non-language-specific rules. Roughly, the mechanism is as follows: each slash in a functional category carries a feature (mode) saying whether it can serve as input to harmonic composition, or crossed composition, or none, or both. This account is perceived as a cleaner solution because it bundles all language-specific aspects of a grammar in one place: the lexicon.

Type-raising is typically subjected to certain restrictions preventing its infinite recursion (Steedman, 2001, Section 3.5).

In statistical parsing, preventing ungrammatical analyses is less of a concern than in descriptive grammars, partly because they are concerned with parsing and not generation, partly because even with properly restricted grammars, the number of theoretically possible analyses of real sentences is huge and a statistical model is needed for finding the presumably intended one(s) anyway. The concern here is more to keep ambiguity manageable for efficient parsing, and this concerns all rule instances, regardless of grammatical constraints.

Grammars for statistical parsing therefore often take rule restrictions to what Baldridge and Kruijff (2003) describe as the "most extreme

case": a finite set of permitted rule instances with specific categories, typically those seen in the training data with a certain minimum frequency. This has been found to speed up parsing at no loss in accuracy (Clark and Curran, 2007; Lewis and Steedman, 2014). Although the grammar is then "only" a context-free one, each rule instance is still associated with a specific rule schema, retaining semantic interpretations which can encode long-range dependencies, as Fowler and Penn (2010) point out.

2.4.7 Type Changing

Phrases with the same internal grammatical structure can perform different functions in a sentence. For example, prepositional phrases can be used as arguments, but they can also act as modifiers to nouns, verb phrases or sentences. Adjectives can be predicates or noun modifiers. Nouns can be noun modifiers in compound nouns, and noun phrases can be noun phrase modifiers in apposition constructions. Participal VPs can act as noun modifiers (as reduced relative clauses) and as nouns (in nominalization). Noun phrases can act as sentence modifiers. Mass and plural nouns can act as NPs by themselves, without any determiner to turn them into such. The distinction between internal structure and external function is known as *constituent type* vs. *constituent function* (Honnibal, 2010, Chapter 3).

There are two major ways of analyzing constituents with potentially multiple functions in CCG: through lexical ambiguity or through type-changing rules. With lexical ambiguity, words with the same type can have multiple lexical entries, one for each possible function. For example, prepositions appear with category PP/NP for argument PPs, $(N\backslash N)/NP$ for noun-modifying PPs, $((S\backslash NP)\backslash(S\backslash NP))/NP$ for VP-modifying PPs, and so on. Adjectives have the category $S[\mathrm{adj}]\backslash NP$ for predicative use, N/N for attributive use, and so on. Nouns, besides N, also have category N/N, for when they appear as a modifier in a noun-noun compound. Not only do these words need multiple categories, but consequently so do their modifiers. For example, an adverb modifying adjectives needs two categories: $(S[\mathrm{adj}]\backslash NP)/(S[\mathrm{adj}]\backslash NP)$ for when the adjective is predicative, $(N/N)/(N/N)$ for when it is attributive. Sim-

ilarly, to allow arbitrarily branching noun-noun compounds, infinitely many noun categories N, N/N, $(N/N)/(N/N)$, $((N/N)/(N/N))/((N/N)/(N/N))$... are needed. The result of this proliferation of modifier categories is a large lexicon that poorly captures the generalizations applying to constituents with the same type.

Type-changing rules, on the other hand, allow a constituent type to have a single category, regardless of its function. Once all modifiers have applied, a type-changing rule can turn the category of the constituent into the required one. Each type-changing rule needs a distinct interpretation (Bos et al., 2004), which can often be given in terms of the interpretations of function words. For example, type-changing rules dealing with reduced relatives can be interpreted like a relative pronoun followed by a passive auxiliary, and the type-changing rule turning a bare noun into a noun phrase has the same semantic effect as the indefinite article:

(11) *Type changing (*)*
$S[\mathrm{ng}]\backslash NP{:}f \;\Rightarrow\; N\backslash N{:}[\![that]\!]@([\![are]\!]@f)$
$N{:}f \;\Rightarrow\; NP{:}[\![a]\!]@f$
...

An example of a derivation with type changing is given in Figure 2.8.

The CCGbank flavor of CCG (Hockenmaier and Steedman, 2007), which we adopt in this thesis, uses a mixed approach. It treats prepositional phrases, adjectives, compound nouns and apposition through lexical ambiguity, with some compromises made to keep the lexicon finite and lexical ambiguity manageable. For example, compound nouns are always analyzed as right-branching, even where this is semantically inadequate. On the other hand, reduced relative clauses, nominalization and bare NPs are handled through type changing. This provides a reasonable trade-off between category proliferation and overgeneration. Honnibal (2010) presents an alternative, unified approach using *hat categories*: complex categories that capture *both* constituent type and constituent function. Statistical parsing experiments with hat categories proved successful, but the approach currently lacks the support of mature tools and resources.

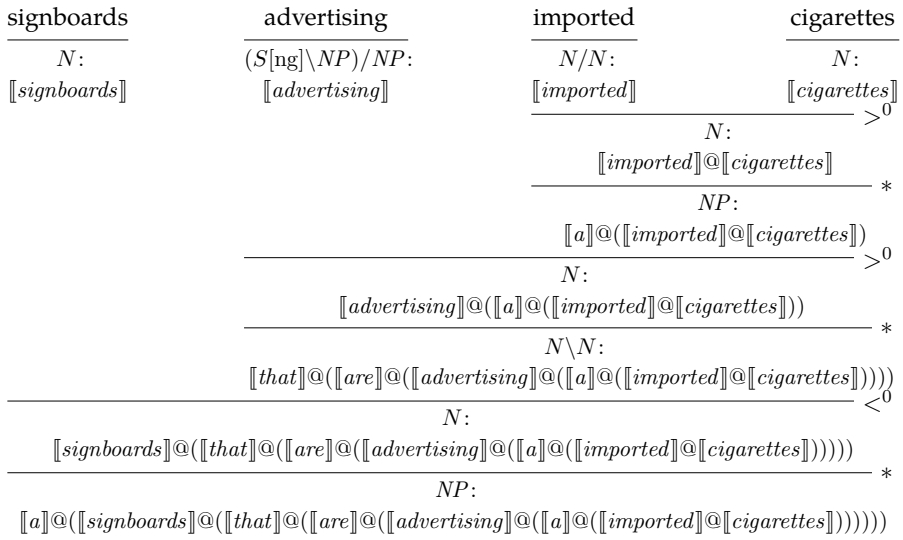


Figure 2.8: Derivation using $N \rightarrow NP$ and $N \rightarrow N\backslash N$ type-changing rules.

2.5 Grammars and Derivations

Let us now make explicit what is a valid CCG derivation. We formally define a CCG as a triple $\langle L, U, B\rangle$ where U is a set of unary rule instances, B is a set of binary rule instances and L is a *lexicon*, i.e., a set of lexical entries of the form $w := C : I$, associating words with possible categories and interpretations. U and B may be infinite, containing all the rule instances permitted by the schemas, possibly limited by rule restrictions, or they may be finite and contain only a fixed set of rule instances, as is common in statistical parsing.

For example the minimal CCG needed for the derivation in Figure 2.3 is

$$
\begin{aligned}
G \;=\; & \langle \{\text{John} := NP{:}[\![John]\!], \\
& \text{wants} := (S[\text{dcl}]\backslash NP)/(S[\text{to}]\backslash NP){:}[\![wants]\!], \\
& \text{to} := (S[\text{to}]\backslash NP)/(S[\text{b}]\backslash NP){:}[\![sing]\!], \\
& \text{sing} := S[\text{b}]\backslash NP{:}[\![sing]\!], \\
& \text{happily} := (S\backslash NP)\backslash(S\backslash NP){:}[\![happily]\!]\}, \\
& \emptyset, \\
& \{S[\text{b}]\backslash NP{:}a \;\; (S\backslash NP)\backslash(S\backslash NP){:}f \Rightarrow S[\text{b}]\backslash NP{:}f@a, \\
& (S[\text{to}]\backslash NP)/(S[\text{b}]\backslash NP){:}f \;\; S[\text{b}]\backslash NP{:}a \Rightarrow S[\text{to}]\backslash NP{:}f@a, \\
& (S[\text{dcl}]\backslash NP)/(S[\text{to}]\backslash NP){:}f \;\; S[\text{to}]\backslash NP{:}a \Rightarrow S[\text{dcl}]\backslash NP{:}f@a, \\
& NP{:}a \;\; S[\text{dcl}]\backslash NP{:}f \Rightarrow S[\text{dcl}]{:}f@a\}\rangle
\end{aligned}
$$

A valid derivation of a linguistic expression w under the CCG is then defined as a directed rooted tree where vertices are constituents, such that

1. each constituent is labeled with a tuple $\langle l, r, C, I\rangle$ where $\langle l, r\rangle$ is called the *span* of the constituent, C is its category and I its interpretation,
2. for each word with position i, there is exactly one leaf in the derivation with span $\langle i, i\rangle$, and there are no other leaves,
3. for every leaf labeled $\langle i, i, C, I\rangle$ where w_i is the i-th word in w, $w_i := C{:}I \in L$,
4. for every internal vertex labeled $\langle l, r, C, I\rangle$, it has either
   - one child, which is labeled $\langle l, r, C_1, I_1\rangle$ such that $C_1 : I_1 \;\Rightarrow\; C{:}I \in U$, or
   - two children, which are labeled $\langle l, m, C_1, I_1\rangle$ and $\langle m, r, C_2, I_2\rangle$, respectively, and $C_1{:}I_1 \; C_2{:}I_2 \;\Rightarrow\; C{:}I \in B$.

A derivation can contain several occurrences of the same category. For the projection operations we define in Chapter 6, it will be useful

to have a notion of which occurrences are *necessarily* identical because their identity is required by the combinatory rules used. We will call those necessarily identical categories *structure-shared*. For example, in Figure 2.2, the modifier *happily* has the category $(S\backslash NP)\backslash(S\backslash NP)$, with two occurrences of the category $S\backslash NP$. By backward application, the first occurrence is structure-shared with the category of *sings*, and the second one with the category of the result constituent *sings happily*. Formally, structure-sharing in a derivation is the least symmetric and transitive relation such that two occurrences of the same (sub)category anywhere in the derivation are in this relation if they are bound by the same variable of a combinatory rule in condition 4 above.

2.6 CCG as a Formalism for Statistical Parsing

Automatic Natural Language Understanding (NLU) is one of the holy grails of Artificial Intelligence. Syntactic parsing has long been seen as an important prerequisite to NLU and consequently is one of the most intensely studied problems in the field. However, there is no consensus on how exactly the output of syntactic parsing should help with NLU. The output is typically a phrase-structure tree with nodes labeled according to constituent type, or a dependency tree with words for vertices and labeled edges. For NLU, we would expect logical formulas that can be interpreted according to some model of the world. Predicate-argument relations, at least local ones, can typically be read off parser output with relative ease, but they only give an incomplete approximation to the meaning, missing aspects such as negation, disjunction, quantification or projection.

Most answers to this problem take a compositional approach. They define functions that recursively interpret each node of a phrase-structure or dependency tree, i.e., map it to a formula, by taking the interpretations of its children as input. Examples of such approaches outside of CCG are Copestake and Flickinger (2000) within the HPSG grammar formalism and Hautli and King (2009) within LFG. Approaches differ, among other things, in how many syntactic rules there are. This has consequences for how many possible local tree configurations the inter-

pretation function has to be defined for, and how strongly this definition depends on the natural language being analyzed.

CCG takes a strongly lexicalized approach here: rules are few and very general, and therefore many interpretation decisions are pushed to the lexical level. For example, information about the syntactic valency of a lexical item and the semantic roles it assign to its arguments are associated with the lexical item itself. At the same time, the way CCG handles non-local dependencies still allows the lexicon to be very compact—for example, the transitive verb *bought* has the exact same category and interpretation no matter whether it appears in a normal clause as in Figure 2.1, as a conjunct as in Figure 2.5 or in an object relative clause as in Figure 2.7.

This simplicity of CCG's approach to compositional semantic interpretation made it an attractive formalism for the output by statistical parsers. Early work on such parsers (Villavicencio, 1997; Doran and Srinivas, 2000) relied on existing hand-written grammars for other formalisms and (semi)automatically translated them to CCG lexicons. As is typically the case with hand-written grammars, these were not robust, i.e., they lacked "the ability to parse real world text with significant speed, accuracy, and coverage" (Hockenmaier et al., 2000). As with parsers for other formalisms, robustness was achieved by training statistical models on large amounts of real world text, annotated with gold-standard parses. The statistical information can then be used to automatically choose the presumably correct parse among the exponentially many possible ones.

The first such statistical CCG parser was presented in Hockenmaier et al. (2000), although not rigorously evaluated. It made use of an algorithm that converts phrase-structure trees from the 1-million-word Penn Treebank corpus (Marcus et al., 1993) to equivalent CCG derivations for training. The resulting dataset was later released as CCGbank (Hockenmaier and Steedman, 2007) and served as training corpus for a series of further work on CCG parsing. Borrowing techniques that had been shown to work well for conventional phrase-structure and dependency parsing, it gradually managed to catch up with them in terms of accuracy, while outputting interpretable CCG derivations. All these parsers focus on syntax, leaving interpretation to external components.

Clark et al. (2002) use a (deficient) probabilistic model that factors the probability of a sentence into the probability of the lexical category sequence and the set of dependencies—roughly speaking, the pairs of argument slots and the heads of the constituents filling them. By modeling the probabilities of dependency structures rather than derivations, they avoid spreading the probability mass too thinly among semantically equivalent derivations, instead modeling all semantically equivalent derivations as the same event.

Hockenmaier and Steedman (2002); Hockenmaier (2003a,b) define a variety of generative, PCFG-like models for CCG derivations. Here, spurious ambiguity is dealt with by training exclusively on derivations that are in *normal form* (Eisner, 1996), i.e., that use composition and type-raising only when necessary. In this way, probability mass is concentrated on normal-form derivations and pulled from non-normal-form semantically equivalent derivations.

Clark and Curran (2004, 2007) train a *discriminative* model, resulting in the first CCG parser to achieve state-of-the-art results in terms of dependency recovery. An important ingredient here, as for Clark et al. (2002), is the *supertagger* component, which finds the most likely category sequences independently of further constituents, thereby cutting down the search space for the parser and dramatically reducing space and time requirements for training and decoding. Another important ingredient is a large number of features included in a log-linear model.

Fowler and Penn (2010) point out that state-of-the-art CCG parsers are in fact context-free, and that this can be exploited by straightforwardly combining them with techniques independently developed to improve the accuracy of context-free parsers. They do this by applying the unsupervised grammar refinement approach developed for context-free parsers by Petrov and Klein (2007), and show that this can improve parsing accuracy.

Auli and Lopez (2011) present further improvements achieved by using task-specific loss functions instead of maximizing log-likelihood, and incorporating supertagging features into the parsing model.

Zhang and Clark (2011) present the first transition-based CCG parser, a shift-reduce parser with a simple perceptron model and beam search. It models (normal-form) derivations in terms of a series of steps

to build them, where the words are processed incrementally from left to right. This parser outperforms some of the best previous chart-based ones while relying less on the supertagger component, in that beam search enables it to take more candidate categories for each token into account. Xu et al. (2014) improve upon this model by modeling dependencies instead of derivations, treating derivations as latent. Ambati et al. (2015) build upon it to create a strictly incremental CCG parser by adding additional actions allowing new material to combine with constituents that have already been reduced.

Lewis and Steedman (2014) present a model that fully reduces CCG parsing to supertagging: it only models the probability of the lexical category sequence, and the parse with the best such sequence is found deterministically using an A* algorithm. The model is simpler and faster than previous chart-based ones, and accuracy is very close to the best reported previous results.

Finally, two recent papers report improved results using neural networks: Xu et al. (2016) combine a shift-reduce CCG parser with a Recurrent Neural Network (RNN) for making transition decisions and task-specific loss similar to Auli and Lopez (2011). Lewis et al. (2016) use the A* parser of Lewis and Steedman (2014) together with a Long Short-term Memory (LSTM) with semi-supervised training.

Chapter 3

Semantic Parsing

3.1 Introduction

In a broad sense, a semantic parser is any system that takes natural-language (NL) utterances as input and outputs formal representations of their meaning or intent (meaning representations, MRs), expressed in some meaning representation language (MRL). Such systems have been around since the 1970s at the latest, see, e.g., Erman (1977) for an early collection of articles or Allen (1994) for a more recent monograph. Traditionally, they relied on *semantic grammars* manually engineered for specific domains. Such grammars require new manual engineering effort for each new domain and tend to be not very robust in the face of the enormous variability of natural language.

Since the 1990s, there has been work on automatizing the construction of semantic parsers using machine learning, with the aim of making them more robust and more easily adaptable to new domains. Since then, the term *semantic parsing* is usually used in a narrower sense, referring to such statistical systems. In this chapter, we review the work that shaped the field of semantic parsing in this narrower sense. Specifically, we limit the scope to work on semantic parsers meeting the following three criteria:

1. The system is *trained*, using machine learning techniques, on training examples. Each training example contains one NL utterance

(e.g., a phrase, a sentence or even a longer text). The reason for this focus on machine learning methods rather than rule-based methods is that our goal in this thesis is to develop methods for cross-lingual semantic parsing that do not require rule writing, and are looking to build on existing machine learning methods for this purpose.

2. If the training examples contain MRs, then those MRs are *not* anchored to the NL utterances. That is, they do not contain explicit information on which parts of the MRs correspond to which substrings of the NL utterance. This allows the MRs a greater degree of abstraction from lexical and syntactic particulars and enables cheaper methods of annotation, but makes the learning task more challenging. Thus, we exclude, e.g., methods relying on parse trees with semantically augmented labels.

3. The method is, at least in principle, able to handle recursive structures, both in the NL utterances and in the MRs. Methods that are not are more aptly described as classification, slot filling, entity linking or relation extraction methods than as semantic parsing methods.

3.2 Early Work

The earliest work on learning semantic parsers was driven by the goal to create natural-language interfaces to database-backed expert systems. Earlier methods to this end had used hand-crafted grammars specific to the respective domain. The desire to avoid hand-crafting and automatically train systems from examples instead motivated a move to more data-driven methods. Much early work focused on the ATIS datasets released as part of a series of challenges organized by ARPA (Price, 1990; Bates et al., 1990). These datasets contain a database of flight information, and exchanges between users and expert systems, where each user request is annotated with an appropriate database query to show the requested information. An example, adapted from Papineni et al. (1997):

(1) a. What are the least expensive flights from Baltimore to Seattle?
b. LIST FLIGHTS CHEAPEST FROM:CITY BALTIMORE TO:CITY SEATTLE

Part of the challenge was to build systems automatically constructing the MRL queries given the NL inputs. Early machine learning methods tackling it did not yet meet our criteria for being called "semantic parsing" because they either required anchored MRs (Pieraccini et al., 1992; Miller et al., 1994) or were incapable of handling recursive structures (Kuhn and de Mori, 1995; Papineni et al., 1997). Recursive structures are not actually required for the task because ATIS utterances are representable by a single frame with slot fillers that are variable in number but flat in structure. He and Young (2003, 2005, 2006) developed a system based on probabilistic push-down automata which does not have these limitations. However, although their system is capable of representing recursive structures and outperforms systems that are not, they still do not test their system on data with MRs complex enough to actually *require* recursive structures.

An important step was the creation of the GEOQUERY corpus (Zelle and Mooney, 1996). This is a dataset of NL queries to a database of United States geographical data, paired with appropriate database queries in a purpose-built MRL. Compared to earlier resources, this one requires quite sophisticated understanding of the hierarchical nature of both NL and MRL expressions, as evidenced, e.g., by recursively nested entity descriptions or "meta-predicates" to compute aggregate values:

(2) a. what is the capital of the state that borders the state that borders texas
b. `(answer (capital (loc_2 (state (next_to_2 (state (next_to_2 (stateid texas:e))))))))`

(3) a. what is the capital of the state with the highest point
b. `(answer (capital (loc_2 (state (loc_1 (highest (place all:e)))))))`

Zelle and Mooney (1996); Tang and Mooney (2001) use a shift-reduce parser to translate NL queries directly into MRL queries. Ambigui-

ties are resolved by rules for choosing parser actions, learned using Inductive Logic Programming. The lexicon for this parser, i.e., word-predicate associations, must be given to the system. This is done manually in their experiments. An automatic acquisition method that can be combined with their systems is presented in Thompson and Mooney (2003). Tang and Mooney (2001) also evaluate their method on the new Jobs dataset, consisting on job postings from a newsgroup together with formal representations.

Another benchmark dataset for semantic parsing is introduced in Kate et al. (2005): the CLang corpus of NL instructions to robotic soccer players in if-then-form, paired with corresponding expressions in a purpose-built MRL. In the following years, the semantic parsing task for both datasets is attacked with a variety of methods, and accuracy is gradually improved.

Kate et al. (2005) present a system that learns to apply a sequence of transformation rules to either a natural-language string or its externally generated syntactic parse tree to transform it into an appropriate MR. The set of rules to use is determined by an iterative procedure trying to cover the training data as accurately as possible. A lexicon is thereby acquired automatically.

The system of Kate and Mooney (2006) learns a probabilistic grammar for the MRL, based on Support Vector Machine classifiers for each production. Terminals are associated with contiguous, not mutually overlapping substrings of the NL expressions and the order of daughters in the parse tree can be permuted to accommodate differing orders between NL and MRL. Ge and Mooney (2005, 2006) and Nguyen et al. (2006) present related approaches making use of trees that describe both the NL and the MRL expressions. The approach however relies on anchored MRs in training.

Wong and Mooney (2006, 2007) also use a formalism describing NL and MRL synchronously, but overcome the need for manual syntactic annotation by using word alignment techniques developed for machine translation. They align NL words to MRL productions, induce a Synchronous Context-free Grammar (SCFG) translating between NL and MRL expressions and train a probabilistic model for parsing with it. In addition to the previous datasets, they also report results for a multilin-

gual subset of GEOQUERY where the NL utterances have been translated to Japanese, Spanish and Turkish.

Lu et al. (2008) define a generative model of NL-MRL expression pairs with *hybrid trees* as latent variables. Hybrid trees are similar to SCFG derivations but are "horizontally Markovized". This makes them more robust in the face of unseen productions required for parsing the test data.

Ge and Mooney (2009) combine an existing syntactic parser, semantic lexicon acquisition through word alignments and a feature-based disambiguation model to derive MRs. Jones et al. (2012) learn synchronous models of NL and MRL using Bayesian Tree Transducers. Finally, Andreas et al. (2013) show that standard machine translation techniques (a phrase-based model and a hierarchical model) can be applied to the task of semantic parsing, treating MRs as target-language utterances, with competitive accuracy.

The first CCG-based approaches to semantic parsing were developed in parallel. Due to their lasting impact on the field and their importance for this thesis, we describe them separately in the next section.

3.3 Early CCG-based Methods

Zettlemoyer and Collins (2005) present the first CCG-based approach to learning semantic parsers. Candidate lexical entries are created by a fixed number of templates, in which placeholders are filled with (multi)-words and non-logical symbols from training examples. The templates encode some grammatical expert knowledge on which categories are needed for the given natural language, and which shapes the interpretations for each category can take. On the other hand, they do not encode any prior knowledge on possible word-template or word-symbol associations, so apart from good entries as in (4), many bad entries as in (5) are generated (examples from their paper):

(4) a. Utah $:= NP : utah$
b. Idaho $:= NP : idaho$
c. borders $:= (S\backslash NP)/NP : \lambda x.(\lambda y.next\_to(y, x))$

(5) a. borders $:= NP{:}idaho$
b. borders Utah $:= (S\backslash NP)/NP{:}\lambda x.(\lambda y.next\_to(y, x))$

A learning procedure then jointly attempts to reduce the lexicon to the good entries and learn parameters for a log-linear model, estimated using stochastic gradient descent, that discriminates between more and less likely derivations for an NL utterance. Additional expert knowledge is added in the form of an initialization of the model so that known good lexical entries for proper names start out with higher-valued parameters. Learning alternates between updating the lexicon so that it only contains the lexical entries needed for the currently highest-scoring derivations, and re-estimating the parameters on the training data using the updated lexicon, treating the derivations as hidden variables. At the time, the method achieved the highest precision and comparable recall to the previous state of the art on the GEOQUERY and JOBS data.

Zettlemoyer and Collins (2007) replace the batch learning algorithm with an online learning algorithm that uses simple perceptron updates instead of Stochastic Gradient Descent. One training example is considered at a time, and both the lexicon and the parameters are updated after each example. Additionally, they introduce a number of type-changing rules (cf. Section 2.4.7) designed to deal with spontaneous, unedited language. For example, "role-hypothesizing" rules can turn the category and interpretation of a noun phrase like *Boston* into that of a prepositional phrase like *from Boston* even if the preposition is missing. Finally, they employ a two-pass parsing strategy where the parser can skip certain words if no parse is found otherwise. With these innovations, their parser set a new state of the art on the ATIS data, which have simpler MRs but less controlled NL utterances and are therefore harder to parse than GEOQUERY.

Kwiatkowksi et al. (2010) present another CCG-based approach motivated by the desire to do away with the manually written templates which are specific both to the natural language to parse and to the MRL to parse into. They do this by turning the above approach on its head: instead of generating candidate lexical entries and learning by trying to build derivations with the correct interpretations, they start by generating a single lexical entry $x := S{:}z$ for every training example, where x

is the entire sentence and z the associated MR. An online learning algorithm then iteratively makes the lexicon more general and jointly learns parameters for a log-linear parsing model. At each step, the algorithm finds the best derivation for a training example according to the current model, then considers a number of variations of this derivation, each variation making a single change: either some non-lexical constituent is replaced by a single multiword constituent with the same category and interpretation, or some lexical constituent H spanning more than one word is split into two constituents F and G. In the latter case, the categories and interpretations of F and G are chosen so that they can combine into H via a combinatory rule, subject to some constraints that limit the splitting possibilities to a finite and polynomial number. The variation resulting in the greatest increase of parse probability according to the current model is chosen, and the corresponding new lexical entries are added to the lexicon. A stochastic gradient update is performed for the training example, and the algorithm moves on the next one. A two-pass parsing strategy similar to that of Zettlemoyer and Collins (2007) is employed. Again, prior knowledge about proper names is given to the system in the form of lexical parameter initializations. Competitive results are reported for GEOQUERY, on the full set with English as NL as well as on the subset with Japanese, Spanish and Turkish translations.

Kwiatkowski et al. (2011) extend the approach by learning a *factored* lexicon. That is, instead of storing complete lexical entries, *lexemes* (pairs of (multi)words and lists of non-logical symbols) are stored separately from *lexical templates* (functions that map lists of non-logical symbols to lexical interpretations containing these symbols). This way, the system can learn more from less data by exploiting systematic variation in word usage. For example, having acquired the lexical entries in (6), the parser can directly use the one in (7), despite never having explicitly added it to the lexicon, by using the lexeme from (6-a) and the template from (6-b) (examples adapted from their paper):

(6) a. flight := $N{:}\lambda x.\mathit{flight}(x)$
b. fare := $N/(S\backslash NP){:}\lambda f.\lambda x.\mathit{cost}(x) \wedge f(x)$

(7) flight := $N/(S\backslash NP){:}\lambda f.\lambda x.\mathit{flight}(x) \wedge f(x)$

The algorithm is also extended with a step that can produce lexical templates "with content". When instantiated with a lexeme, these templates introduce additional non-logical symbols regardless of the lexeme. Such templates play the same role as the special type-changing rules of Zettlemoyer and Collins (2007), allowing, e.g., to interpret the noun phrase *Boston* like the prepositional phrase *from Boston*. They are, however, not manually specified, but learned from the training data using certain heuristics. They report highly competitive results on the GEOQUERY dataset and also successfully apply their approach to the ATIS dataset.

Despite approaching the generation of lexical entries "from opposite ends", the approaches of Zettlemoyer and Collins (2005, 2007) on one hand and Kwiatkowksi et al. (2010); Kwiatkowski et al. (2011) on the other hand are actually quite similar: both acquire a CCG lexicon and a log-linear model scoring derivations by learning from NL utterances paired with MRs, treating CCG derivations as hidden variables. In fact, both approaches have been formulated within a single algorithmic framework and implemented withing a single software framework (Artzi and Zettlemoyer, 2013a).

3.4 Context-dependent and Situated Semantic Parsing

A given NL utterance may not have the same MR every time it is encountered. Its correct interpretation may depend on additional information available to the system. This additional information may be a model of the current state of the world which may change after each utterance as a result of the system taking actions based on the derived MR. In this case the semantic parsing task is called *situated*. It may also depend on the preceding discourse, i.e. previously encountered NL utterances and their interpretation. In this case, the semantic parsing task is called *context-dependent*. An example of a context-dependent semantic parsing task is the interpretation of user follow-up requests within human-machine interactions in the ATIS dataset. Here is an example (from Zettlemoyer and Collins (2009)) of an initial request and a follow-

up request, each annotated with the correct MR (a database query written in λ-form, selecting a list of flights to show). The interpretation of the follow-up request clearly depends on the preceding dialog (which forms the context in this case), as its MR contains some material from the initial request:

(8) List flights to Boston on Friday night.
$\lambda x.flight(x) \wedge to(x, bos) \wedge day(x, fri) \wedge during(x, night)$
Show me the flights after 3pm.
$\lambda x.flight(x) \wedge to(x, bos) \wedge day(x, fri) \wedge depart(x) > 1500$

Zettlemoyer and Collins (2009) present a system for this task that parses NL utterances into underspecified, context-independent MRs and then uses a learned weighted linear model to choose among the various possibilities defined by the context of expanding the underspecified MRs to full MRs. In Vlachos and Clark (2014), a similar dataset and imitation learner for semantic parsing is presented.

In these systems for context-dependent semantic parsing, the parser has access to the full MRs during training. In situated semantic parsing, MRs are typically not provided for training, and the learner has to figure them out via interaction with the model of the world. We will encounter such approaches in the following section.

3.5 Alternative Forms of Supervision

With semantic parsers achieving high accuracies on the ATIS, GEOQUERY and CLANG datasets, much subsequent work focused on learning semantic parsers more cheaply, i.e., with less explicit supervision. While existing methods did not require anchored MRs, they still required each training NL utterance to be annotated with a full MR in the target MRL, something that still takes considerable expert knowledge and effort to do. Learning algorithms powerful enough to learn as much from less explicit annotation were desired.

3.5.1 Ambiguous Supervision

Often, the context in which an NL utterance occurs provides clues as to what it might mean. For some applications, the context is informative enough to automatically enumerate a small set of all possible alternative MRs, relieving humans of the annotation burden at the cost of a certain remaining ambiguity. A semantic parser can then be learned from the alternatives, trying to filter out the wrong MRs as noise. A number of approaches tackles this problem with expectation-maximization-like algorithms, i.e., by cycling for a number of iterations between two steps: 1) finding the best correct parses according to the current model and 2) re-estimating the model parameters accordingly.

Kate and Mooney (2007) extend the learning algorithm of Kate and Mooney (2006) to this end. After some rule-based pruning of the ambiguous training data, they first treat all remaining alternatives for a training example as correct and weigh them uniformly for training the SVM-based model. In later iterations, MRs with non-optimal parsing scores for each training example are dropped. They evaluate their approach on two datasets with artificially created ambiguity, in one case randomly and in one case trying to approximate the type of ambiguity that a young child would experience when confronted with utterances in a real-life context. Chen and Mooney (2008) present a similar extension to the algorithm of Wong and Mooney (2007) and apply both extended algorithms to a sportscasting dataset where the ambiguity is not artificially generated, but stems from the fact that a human comment may refer to any of a number of recent events in the game.

Kim and Mooney (2010) present a probabilistic generative model, based on that in Lu et al. (2008), which not only produces semantic parses but also segments the stream of NL utterances and discards irrelevant ones. It is learned from the same type of supervision as before, without explicit segmentation or marking of irrelevant utterances in the training data. Börschinger et al. (2011) present a similar approach based on a joint representation of NL and MR structures in a Probabilistic Context-free Grammar (PCFG).

3.5.2 Highly Ambiguous Supervision

We speak of *highly* ambiguous supervision if the number of possible MRs for a given NL utterance is in general too large to explicitly enumerate and/or to tackle with the above methods. This is the case for example when training systems to parse NL navigation instructions into MRL *action plans* and using the paths taken by humans following the same instructions as supervision signal. The number of possible action plans consistent with a given observation is exponential in the length of the path.

In the task considered by Matuszek et al. (2010), this is the case because maps may be incorrectly labeled, hence each landmark reference in an action plan may be incorrect. They solve it by heuristically segmenting NL instruction sequences, parsing the segments with the system of Wong and Mooney (2006) and revising parsing and map labeling decisions when otherwise unable to execute the action plan.

In Chen and Mooney (2011); Chen (2012), the exponential ambiguity in supervision stems from the fact that action plans may or may not contain landmark references at each turn. They tackle the problem by using a separate lexical alignment step to choose the action plan most likely to correspond to the given instruction sequence before training the parser. Since the alignment step sometimes makes mistakes, they choose a semantic parser learning algorithm known to be particularly robust to noise, *viz.* that of Kate and Mooney (2006). Kim and Mooney (2012) combine the lexical alignment step of Chen and Mooney (2011) with the unsupervised PCFG learning approach of Börschinger et al. (2011) to solve the same problem.

3.5.3 Learning from Exact Binary Feedback

The next degree of reduced supervision occurs when the system does not have access to *any* example MRs and can only check whether a given MR leads to the desired effect when executed. That is, training examples contain an NL utterance, some state of the world and a *validation function* that takes an MR and the state and returns true or false depending on whether executing the MR on the state leads to the correct answer. The

learner must essentially proceed by trial and error, hypothesizing full MRs and checking correctness via the validation function.

For example, Clarke et al. (2010) train a parser for the GEOQUERY domain without access to the MR annotation, only to the geography database and the answer expected when querying the database with the correct MR. The MR and alignment of its constituents to NL substrings is modeled as a latent variable. Liang et al. (2011) present a similar system, using a different, dependency-tree-like MRL for latent modeling whose structure makes its easier to align to NL utterances. This system achieved higher accuracies at the time than even the best more strongly supervised systems, also on the JOBS domain, albeit using a manually crafted seed lexicon.

Goldwasser and Roth (2011) present a system that learns to parse NL descriptions of the rules of a card game into Horn clauses which can determine whether a given move is legal or not. Their validation function applies the hypothesized Horn clause to a small number of positive and negative examples of valid moves and checks whether it permits them or not. They show that the system is able to generalize to both unseen game states and to unseen instructions.

Artzi and Zettlemoyer (2013b) tackle the navigation task previously addressed by Chen and Mooney (2011); Chen (2012); Kim and Mooney (2012), experimenting with different validation functions: one which gives feedback after each step in a sequence of navigation instructions, and one that only validates the final position of the navigation agent. This system is also an instance of a *situated* semantic parser, taking environment cues into account for making parsing decisions such as word sense disambiguation.

3.5.4 Learning from Approximate and Graded Feedback

In some tasks, not even certain information on whether a given MR for a given training example leads to the correct result when executed is available. Sometimes the learner can then draw on heuristic and/or approximate reward functions that may sometimes make mistakes but on the whole still provide feedback valuable enough for successful learning.

For example, Branavan et al. (2010) present a system that parses high-level instructions to Windows users ("open control panel") into low-level command sequences ("left-click Start, left-click Settings, left-click Control Panel"). The reward function approximates correctness by a simple heuristic criterion: every NL sentence refers to at least one command and has words in common with the label of the environment object the command refers to, e.g., the "Start" button.

Artzi and Zettlemoyer (2011) generalize the framework of Zettlemoyer and Collins (2005, 2007) using a loss-sensitive perceptron algorithm which allows for custom reward functions (here: loss functions) returning real-valued measures of MR "correctness". They apply it to the scenario where a dialog system did not understand the user's initial request and so followed it up with a series of clarification questions and confirmation statements. The log of this further dialog is then used to supervise learning to parse the initial request via a custom loss function.

3.5.5 Unsupervised Learning

A special case is the unsupervised learner of Goldwasser et al. (2011), which trains a GEOQUERY parser with the objective of optimizing the overall confidence of the model over all (unlabeled) training examples. They start with the observation that certain repeating patterns in the parses predicted by a model are an indication of high model quality because they typically capture real regularities in the data. They estimate confidence by a number of measures capturing such patterns.

3.6 Scaling Semantic Parsers to Larger Vocabularies Using Two-stage Methods

One thing most of the semantic parsing methods mentioned above have in common is that they learn to interpret words and syntactic constructions from a single set of training examples. This makes sense for small domains like GEOQUERY, where both the number of syntactic constructions and the number of distinct non-logical symbols in the MRL are rather small. However, it is problematic for larger domains as found

in general-purpose information systems like Freebase (Bollacker et al., 2008). Here, the number of non-logical symbols (representing concepts, relations and individuals) and NL (multi)words that might refer to them is orders of magnitude higher, while a small number of syntactic constructions still suffices to parse the vast majority of relevant NL utterances, such as factoid questions. Learning the constructions thus needs relatively few training examples, but is complex and difficult to do without explicit supervision. The learning of word meanings on the other hand needs much more data but can be done with less supervision and less complex algorithms. Accordingly, a major recent trend in semantic parsing is to (partially) separate the acquisition of word meanings from the acquisition of parsing models, both in terms of the data used and in terms of the algorithms. We refer to such methods as *two-stage methods*.

3.6.1 Lexicon Generation and Lexicon Extension

A first family of two-stage methods explicitly acquires or extends a lexicon for the semantic parser. The process is driven by the knowledge base (KB) which defines the non-logical symbols that the MRL uses, typically Freebase. The KB contains a number of known instances of concepts (e.g., City(Sacramento)) and relations (e.g., LocatedIn(Sacramento, California)). Large amounts of texts, such as the World Wide Web, can then be mined for sentences in which the entities in known concept and relation instances are referenced (this is often easy to do with simple string matching or separately developed mention detection techniques), e.g., the sentence *Sacramento is a city in California.* The sentence can then be used to generate candidate lexical entries, e.g., associating the word *city* with the concept City and the word *in* with the relation LocatedIn.

Krishnamurthy and Mitchell (2012) first acquire the lexicon in this way, using an existing syntactic dependency parser to parse the sentences retrieved and extracting CCG lexical entries from the resulting dependency trees using hand-written rules. They then train a probabilistic CCG parser in a weakly supervised fashion, with the global objective of making the interpretations of retrieved sentences a "good description" of the knowledge base in the sense that "every relation in-

stance is expressed by at least one sentence", along with a syntactic objective that uses the external syntactic parser as supervision. This training process also has the function of telling good candidate lexical entries from bad ones.

Cai and Yates (2013) use a semi-supervised approach on FREE917, a newly introduced dataset of factoid questions such as *What movies did Steven Spielberg direct?*, annotated with MRs. They first train the system of Kwiatkowksi et al. (2010) on a small number of question-MR pairs. They then extend the resulting Probabilistic CCG with additional lexical entries for words not found in the supervised training data. Entries are automatically generated using the initial entries as templates, and likely correct word-symbol pairs are identified by a regression model. This model aggregates various association statistics such as Pointwise Mutual Information and is trained on the small training dataset, which was manually annotated with symbol-word associations.

Berant et al. (2013) generate candidate lexical entries for the DCS formalism (Liang et al., 2011) using *alignment*, a similarity measure between words and symbols in terms of their *extensions*, i.e., sets of co-occurring entity pairs. Additionally, they introduce a *bridging* operation, exploiting the type system of Freebase to hypothesize matching roles where no words are found that explicitly express them. As with Krishnamurthy and Mitchell (2012), the selection of good lexical entries is mostly left to the final parsing model. They also introduce a new dataset of factoid question-answer pairs, much larger and more challenging than FREE917: WEBQUESTIONS.

3.6.2 Semantic Parsing via Ungrounded MRs

Another family of two-stage methods is to first parse to an intermediate representation (variously called "quasi-logical form" or "ungrounded" MR) without attempting to map NL words to MRL symbols. The symbols of the ungrounded MR, then, are usually just words. Ungrounded MRs cannot directly be executed. To make them useful, e.g., for question answering, a second stage is employed.

Figure 3.1 compares different approaches to question answering and the role semantic parsing plays in them. Approaches using informa-

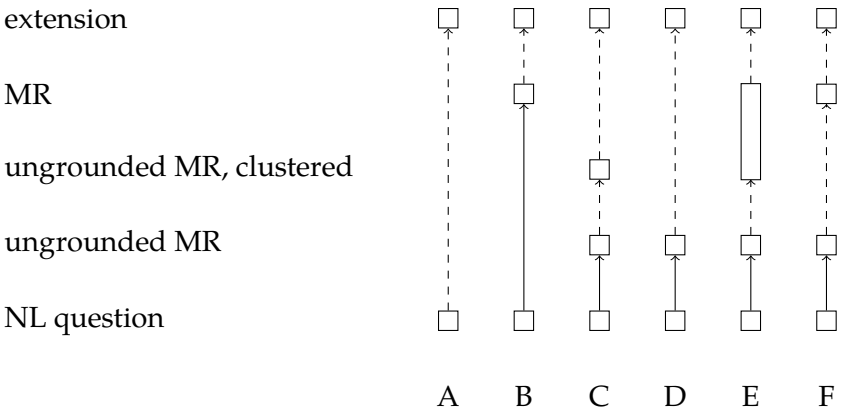


Figure 3.1: The role played by parsing (solid arrows) and formal meaning representations in various approaches to question answering: information extraction methods (A), classical semantic parsing (B), ungrounded unsupervised semantic parsing (C), open-vocabulary semantic parsing (D), grounded unsupervised semantic parsing (E), semantic parsing with ontology matching (F).

tion extraction techniques (A) do not use semantic parsing and do not attempt to explicitly represent the meaning of the question, but map questions straight to the answer. Classic semantic parsing methods (B) parse the NL question into an MR, which is then used to query the KB and retrieve an answer.

Unsupervised ungrounded semantic parsing methods (C), again, do *not* attempt to build a (grounded) MR. Therefore, they cannot be used to query formal KBs. However, they are useful to query large amounts of *text*, used as an "unstructured knowledge base". To this end, they parse NL utterances into ungrounded MRs (which can just be syntactic dependency trees), then attempt to reduce semantically equivalent but syntactically different parts of the MR (e.g., nodes or fragments of it) to the same representation by clustering them. Once both the text and questions are represented as clustered ungrounded MRs, question answering can be done by matching the latter against the former. Poon

and Domingos (2009) perform the clustering using Markov Logical Networks, and Titov and Klementiev (2011) using a Bayesian model. Both evaluate their models on a question answering task using biomedical text as a textual knowledge base.

Krishnamurthy and Mitchell (2015) apply a similar method, under the label of "open-vocabulary semantic parsing" (D), to the larger domain of Freebase entities. They do not query Freebase itself, instead they query a large amount of automatically entity-linked (but otherwise unannotated) text. They use a CCG parser to generate ungrounded MRs for the text, which serve as a collection of "observed facts". They then train a Probabilistic Database to rank (question, answer) pairs that correspond to observed facts over pairs that do not. Instead of trying to reduce semantically equivalent ungrounded MRs to the same form, the Probabilistic Database scores the ungrounded MR of a question directly. They find that querying text can answer many questions that querying KBs cannot because not all questions are expressible in the KB's MRL. However, for questions that *are* expressible, the KB gives more accurate results. There is thus potential in building systems that can query both text and KBs.

To apply unsupervised semantic parsing techniques to the task of querying KBs, Poon (2013) introduces *grounded* unsupervised semantic parsing (E). This approach tries to put dependency tree nodes into semantically equivalent clusters (here: "semantic states") in an unsupervised fashion, similar to Poon and Domingos (2009). However, the clustering is now restricted to use symbols from the target knowledge base as semantic states so that clustered ungrounded MRs are effectively the same thing as grounded MRs. Learning the correct node-symbol associations is bootstrapped using string matching measures, thus assuming that the KB symbols are made up using words from the same language (here: English) as the NL parsed. Adding mechanisms for handling non-local semantic relations and some rules for domain-independent phenomena like superlatives or the interpretation of numerals, this system achieves performance on the ATIS task effectively tying with supervised methods, despite using neither MR annotations nor gold-standard answers for learning.

A slightly different way of going from ungrounded to grounded

MRs is *ontology matching* (F). Here, the system learns to transform the former into the latter. Kwiatkowski et al. (2013) use a fixed, domain-independent (but language-specific) CCG lexicon for English to construct ungrounded λ-calculus MRs. These are then transformed into grounded MRs using a series of transformation operations which replace subexpressions of the ungrounded MR with shorter expressions, containing the same free variables and introducing new ad-hoc symbols. Finally, the ad-hoc symbols are mapped to KB symbols. As examples, the transformation steps for two subexpressions (from their examples) are shown here:

(9) a. $\iota z.Public(z) \wedge Library(z) \wedge Of(z, NewYork)$
b. $PublicLibraryOfNewYork$
c. `new_york_public_library`

(10) a. $\lambda x.eq(x, count(\lambda y.People(y) \wedge \exists e.Visit(y, \ldots, e) \wedge Annually(e)))$
b. $\lambda x.HowManyPeopleVisitAnually(x, \ldots)$
c. $\lambda x.$`public_library_system.annual_visits`$(x, \ldots)$

The transformation can be seen as making the "overly compositional", syntax-driven analysis provided by the parser less compositional to better fit the knowledge base. It serves roughly the same function as the collapsing of dependency graph fragments and their clustering in Poon and Domingos (2009). Knowledge about likely word-symbol associations is incorporated via rich features drawn from language-specific lexical resources and the KB. Parsing and transformation steps are trained jointly on question-answer pairs using a linear model. Similarly, Reddy et al. (2014) obtain ungrounded MRs from a CCG parser, then ground them by relabeling nodes and edges. The parsing and grounding steps are trained jointly and discriminatively to rank correct (ungrounded, grounded) MR pairs over incorrect ones. The approach does not use question-answer pairs, instead it generates training data synthetically using the KB and a large entity-linked text corpus.

3.6.3 Semantic Parsing or Information Extraction?

The focus on new benchmark datasets like Free917 and WebQuestions has helped semantic parsing techniques scale up to NL fragments with many more word types, and to target MRLs with many more symbols. But the focus on simple factoid questions risks losing sight of the challenges that originally motivated semantic parsing. Question answering can be done without constructing meaning representations, at least when the questions are simple. Yao and van Durme (2014) present one such system targeting the Freebase domain, which uses a syntactic dependency parser but does not do semantic parsing. An analysis by Yao et al. (2014) shows that this system does not significantly underperform when compared to the state-of-the-art semantic parser of Berant et al. (2013), and for the most part learns the same features, despite not containing a semantic parsing model. As a further case in point, the model of Berant and Liang (2014) matched the state of the art accuracy on the Free917 benchmark and vastly advanced it on WebQuestions. Yet, it does not do semantic parsing as such, instead relying on the low structural variability of target MRs—reminiscent of the venerable ATIS requests—to generate a broad set of candidates and score them all. This suggests, as Yao et al. (2014) point out, that the shift of focus from GeoQuery to Freebase factoid questions may have been a step back in terms of structural complexity and semantic parsing techniques are thereby underchallenged. They suggest that the semantic parsing community should focus on utterances and datasets with higher structural complexity. One step into way more challenging territory is broad-coverage semantic parsing, which we will take a look at in the next section.

3.7 Broad-coverage Semantic Parsing

The semantic parsing methods we have seen so far target rather restricted styles of text, e.g., robot instructions or factoid questions. Their motivation, design and evaluation are strongly influenced by concrete tasks such as flight booking or factoid question answering, and MRLs are chosen to fit the needs of the respective task. As we have seen, learning methods that learn to map NL utterances to MRs from training ex-

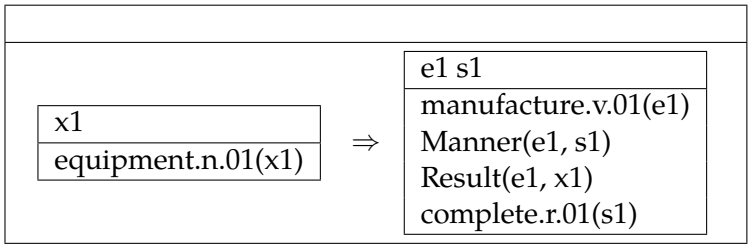


Figure 3.2: DRT-based MR for the sentence *All equipment will be completely manufactured*, adapted from Bjerva et al. (2016).

amples, without manually written semantic grammars, are well established for this type of semantic parsing, which we call *narrow-coverage*.

By contrast, *broad-coverage* semantic parsing targets many styles of NL utterances, including long and complex ones like newswire articles. They are less influenced by specific tasks and more by semantic theory, the desire for a unified representation of many aspects of meaning (see for example Basile et al. (2012b); Banarescu et al. (2013)) and the goal to use Automated Reasoning for Natural Language Understanding (see for example Blackburn and Bos (2005); Bos and Markert (2005)). Accordingly, MRLs are chosen not so much with an eye on any specific task, and more by their adequacy as a general formalism for representing natural-language meaning. As with narrow-coverage semantic parsing, for a long time broad-coverage semantic parsing relied on hand-written semantic grammars. Because of the greater complexity and variability of both targeted NL utterances and MRs, it is not surprising that broad-coverage datasets annotated with MRs are considerably more difficult to create, and that, accordingly, it took longer for learning methods to be applied to broad-coverage semantic parsing.

As an example of an influential broad-coverage system for semantic analysis which maps (syntactic analyses of) sentences to MRs in a rule-based fashion (and is thus not a semantic parser in our narrow sense), consider Boxer (Bos, 2008). Boxer takes CCG derivations without interpretations as input, as produced by a syntactic CCG parser such as

that of Clark and Curran (2007). It assigns every word an interpretation based on its category, part-of-speech tag, lemma and possibly other tags provided by external taggers, such as named-entity taggers or word-sense disambiguation systems. The MRL is based on Discourse Representation Theory (DRT; Kamp and Reyle, 1993). The inventory of non-logical symbols is drawn from WordNet (Fellbaum, 1998) for concepts and VerbNet (Kipper et al., 2008) for predicate-argument relations. By default, concepts and relations are chosen using heuristics and statistics collected offline. An example MR is shown in Figure 3.2.

The release of the Groningen Meaning Bank (Basile et al., 2012b, see also Chapter 5 of this thesis), a large, multi-domain corpus containing sentence-MR pairs produced by Boxer and partially manually corrected, opened the way to applying established learning methods for semantic parsing to a broad-coverage domain. To this end, Le and Zuidema (2012) define a semantic composition formalism in which fragments of the DRT-based MRs are represented as graphs and combined via two operators: *binding* (unifies two discourse referents) and *wrapping* (places a condition into a box). They then use machine learning to acquire a lexicon of suitable fragments and train a probability model guiding the composition of fragments into words. Beschke et al. (2014) present and evaluate a method for semantic CCG induction on the same dataset.

The release of AMRbank (Banarescu et al., 2013) spurred a much larger wave of learning methods now applied to broad-coverage semantic parsing. AMRbank is a large, multi-domain corpus containing sentences manually annotated with AMRs (Abstract Meaning Representations). AMRs are directed acyclic graphs. Non-leaves represent entities, including events. Leaves represent concepts. The concept symbols used are PropBank framesets (Palmer et al., 2005) for events, English words for common (non-event) nouns and Wikipedia URLs for named entities. Edges connect entities to their concepts and events to their participants and are labeled with PropBank argument labels and other relation symbols. The AMR MRL puts a strong emphasis on semantic normalization, reducing syntactically different forms of expressing similar semantic frame instances to the same representation. For example, the phrases "a handsome sailor", "a handsome person who sails" and "a sailor is handsome" all have equivalent AMRs. An example AMR

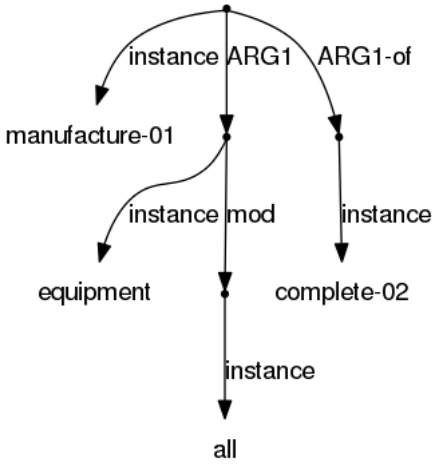


Figure 3.3: AMR for the sentence *All equipment will be completely manufactured*, adapted from Bjerva et al. (2016).

is shown in Figure 3.3. For a discussion of the merits of AMRs as a general-purpose MRL, see Bos (2016).

Flanigan et al. (2014) present JAMR, a system that breaks the problem of mapping NL utterances to AMRs down into two stages: *concept identification*, generating disjoint AMR subgraphs corresponding to content words, and *relation identification*, finding the right relations between them. Concept identification is approached using sequence labeling, and relation identification using a maximum-scoring connected subgraph algorithm. Training relies on an explicit mapping between (multi)words and AMR subgraphs, which is induced automatically using an aligner. Werling et al. (2015) observe that the concept identification stage is the more difficult one and improve it, replacing the static lexicon with a more robust subgraph generation method that captures the regular ways in which AMR maps content words into subgraphs. For example, numerals like *five* are systematically mapped to values like *5*, deverbal nouns like *sailor* often introduce frames like sail-01, etc.

Wang et al. (2015b,a) present CAMR, a system that uses a transition-

based algorithm to transform syntactic dependency trees into AMRs. The transition actions can label nodes, label edges, swap heads and dependents, reattach nodes, introduce re-entrancies, etc. It chooses transformation actions at each stage according to a linear model, trained using the averaged perceptron algorithm. Artzi et al. (2015) train a CCG parser that produces AMRs in two stages. In the compositional first stage, underspecified CCG derivations are created where all relation symbols except ARG0 and ARG1, as well as the entity nodes representing anaphora, for example of pronouns, are replaced by placeholder terms. In the non-compositional second stage, a Factor Graph model resolves anaphoric references and fills in the relation symbols. The initial lexicon for the CCG parser is created by annotating 50 training examples. Learning then alternates, similar to Zettlemoyer and Collins (2005); Artzi and Zettlemoyer (2013b), between updating the lexicon and updating the model parameters.

The SemEval 2016 shared task on AMR parsing brought many more approaches to the scene. Many of them are incremental improvements of CAMR, e.g., adding new features, new lexical resources, neural networks, heuristic post-processing or ensembles. Two participating systems, one of which based on Boxer, are not trained on AMR data but employ independently developed toolchains and conversion of the output to AMR. Novel approaches include one employing a Learning to Search algorithm, a novel transition-based algorithm and one combining the predictions of specialized neural networks with a greedy algorithm. For details, see May (2016).

In Chapter 7, we will add to the work on broad-coverage semantic parser learning by introducing a system that learns via cross-lingual lexicon induction and cross-lingual supervision.

Part II

Narrow-coverage Semantic Parsing

Chapter 4

Situated Semantic Parsing of Robotic Spatial Commands

4.1 Introduction

When interpreting utterances, humans seldom look at the utterances alone. Knowledge about the context in which an utterance is made helps in determining its correct interpretation. This can also be true of computers. Consider the simple example of the robot in Figure 4.1, faced with a board of colored shapes and the instruction "Move the pyramid on the blue cube on the gray one." There are at least two ways to interpret this instruction: "Move the pyramid that is sitting on the blue cube onto the gray cube", or "Move the pyramid that is sitting on the blue cube which is itself sitting on the gray cube." But only the former interpretation makes sense in the pictured situation, because there is no blue cube anywhere on the board sitting on a gray cube.

Using the state of the world (the board in this case) as additional information, a semantic parser should be able to arrive at the correct interpretation more reliably. We investigate this hypothesis in this chapter in the context of the SemEval 2014 Shared Task 6 (Dukes, 2014), a narrow-domain semantic parsing task that consists in mapping natural-language robotic commands to unambiguous commands in a formal language, Robot Control Language (RCL; Dukes, 2013a). In particular,

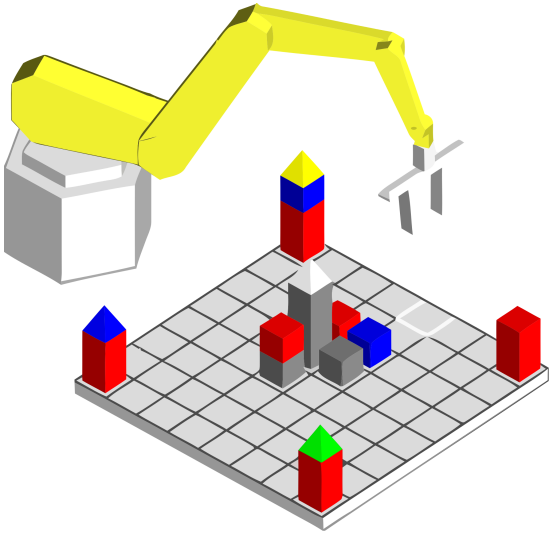


Figure 4.1: A robot faced with the command: "Move the pyramid on the blue cube on the gray one". Picture from Dukes (2014).

we investigate whether existing techniques for semantic parsing with CCG can easily be adapted to this new domain and target representation, and in how far integrating reasoning about the context helps with finding the correct interpretations.

4.2 Task Description

Each instance of the task consists of a context, an English command and an interpretation of the command in RCL. The configuration and the English command are input to the system, and it has to come up with the correct interpretation.

The context is a machine-readable representation of a board consisting of 8x8 tiles. Each tile may hold a shape, which has one of eight colors and is either a cube or a pyramid. Cubes can also support other shapes stacked on them, but pyramids cannot. Each shape on the board must be supported by a cube or a tile, it cannot hover. The context also spec-

ifies the current position of the robot's gripper over the board, and the shape it is currently holding, if any.

RCL expressions are rooted ordered trees whose nodes are labeled with *tags*. We will write them in the form $(t : \mathbf{h})$ where t is the root tag and $\mathbf{h}$ is the sequence of subtrees of the root's children. Leaves are abbreviated as just their tags. For example, here is an English command with an RCL interpretation:

(1) a. move the green pyramid in the bottom left corner

b. $(event:$
$(action:move)$
$(entity:(color:green)\ (type:prism))$
$(destination:(spatial\text{-}relation:(relation:within)$
$(entity:(indicator:back)\ (indicator:left)$
$(type:corner))))$

Subtrees tagged $entity$ denote objects such as shapes, tiles or edges and corners of the board. Their descriptions consist of subtrees tagged $type$ specifying their type, $color$ specifying their color, $spatial\text{-}relation$ describing spatial relations to other objects, etc. Trees tagged $event$ denote actions the robot should take; the action type is specified by subtrees tagged $action$, containing one of three values: $take$ (take an object into the gripper), $drop$ (drop an object that is currently in the gripper onto a specified object) or $move$ (take and then drop an object). Trees tagged $sequence$ denote sequences of events, i.e., instructions that the robot is supposed to carry out one after the other. For a more detailed and complete description of RCL, the reader is referred to Dukes (2013a).

Train Robots (Dukes, 2013b) is a dataset consisting of several thousand such instances. The English commands were collected through crowdsourcing and translated to RCL through manual expert annotation. The pre-terminals (parents of leaves) of RCL trees also contain links to corresponding English words in the command where applicable.

For the SemEval shared task, participants were provided with 2,500 instances for training and development. Final evaluation was performed on a held-out set of 909 instances. A *spatial planner*, a component that

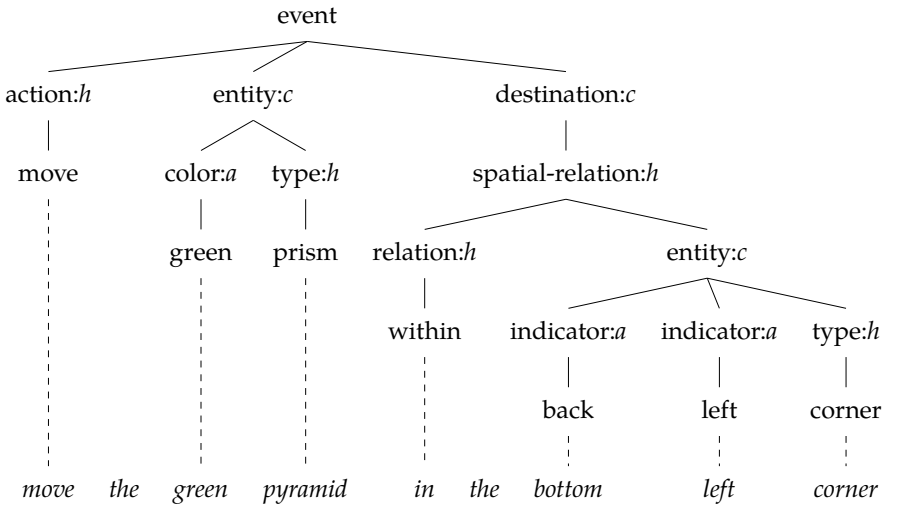


Figure 4.2: The RCL expression from Example (1) represented as a tree diagram. Internal nodes are annotated with constituent types h, a and c. Pre-terminals are aligned to words in a corresponding natural-language expression.

takes RCL expressions and creates execution plans for the given context, was provided as a Java library.

4.3 Extracting a CCG from RCL

4.3.1 Transforming RCL Trees to CCG Derivations

To apply CCG parsing techniques to the task, we first need a corpus of CCG derivations of English commands whose interpretations are RCL expressions. We obtained this corpus by automatically transforming RCL expressions into derivations. To make this mapping more direct and unambiguous, we do not use CCG's standard basic categories N, NP, S and PP, but instead basic categories corresponding to RCL tags.

In each training example, each pre-terminal (parent of a leaf) can be aligned to one or more words in the corresponding natural language

expression. An example is shown in Figure 4.2. Since the alignments to words are not crossing, we can interpret the RCL tree as a phrase structure tree for the sentence and use the algorithm of Hockenmaier and Steedman (2007) to transform it into a CCG derivation. We extend the algorithm with a semantic step that makes sure the derivations would produce the original RCL expressions. The procedure is as follows:

1. Determine constituent types. Each subtree is assigned a constituent type depending on its tag. We treat $action$, $relation$ and $type$ constituents as *heads*, $entity$ and $destination$ constituents as *complements* (i.e., arguments) and $cardinal$, $color$, $indicator$, $measure$ and $spatial\text{-}relation$ constituents as *adjuncts* (i.e., modifiers). For $sequence$ nodes that have multiple $event$ children, we treat the first as head and the rest as adjuncts. A corresponding *constituent type label* h, a or c is added to the label of each internal node (cf. Figure 4.2).

2. Assign lexical semantics. To the label of each pre-terminal, an RCL expression is added which is a copy of a connected subgraph of the tree itself (without the constituent type labels). Which subgraph precisely depends on the type of pre-terminal: For *a*-type and *c*-type nodes, the subgraph includes only the pre-terminal and its daughter. For *h*-type nodes the parent is also included, as well as any subtrees tagged id or $reference\text{-}id$ the parent may have (these subtrees represent anaphoric links and are not linked to other lexical material). To illustrate, the label of the action:*h* node in Figure 4.2 becomes action:*h*:$(event:(action:move))$, and color:*a* becomes color:*a*:$(color:green)$. The leaves are now no longer needed, so we remove them, making the pre-terminals the new leaves. Remember that words are linked to the pre-terminals, not terminals, so the links are not lost.

3. Add sequence nodes. RCL expressions that contain more than one event have a root node tagged $sequence$, which is the parent of all $event$ nodes. To preserve this structure in the CCG transformation, an additional sequence node tagged $sequence$ is introduced between the root and the child, with the same constituent type and semantics as the child.

4. Binarize the tree. Each local tree with more than two daughters is binarized by inserting dummy nodes, provisionally labeled $C:h$ where C is the tag of the parent. The order of binarization is determined by the

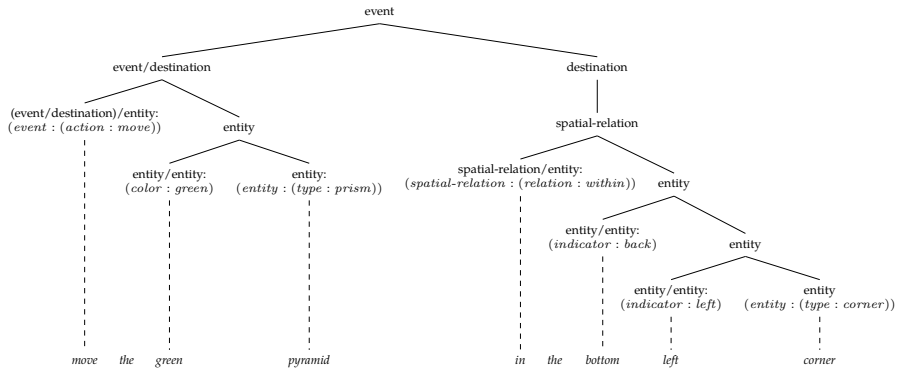


Figure 4.3: Result of the CCG transformation applied to the tree in Figure 4.2.

constituent types: left adjuncts (such as the first *indicator* in Figure 4.2) are split off first, followed by right adjuncts (such as the *destination* in Figure 4.2), left complements and right complements.

5. Assign CCG categories. Starting from the root, the tag of each node is replaced by a CCG category:

- The root gets its tag (*event* or *sequence*) as category.
- *c*-type nodes get their tag (*entity* or *destination*) as category. Their sibling gets category P/T if it is on the left and $P\backslash T$ if it is on the right, where T is the tag of the *c*-type node and P is the category of the parent. For example, the *destination* node in Figure 4.2 gets *destination* as category, and its left sibling therefore gets $event/destination$ because the parent's category is *event*.
- *a*-type nodes such as the two *indicator* nodes in Figure 4.2 get category P/P if they are on the left of their sibling and $P\backslash P$ if they are on its right, where P is the category of their parent. The sibling gets category P.
- Internal nodes without siblings get their tag (*entity* or *spatial-relation*) as category.

Constituent type labels are dropped. The result for our example is shown in Figure 4.3.

4.3.2 The Lexicon

Once the training corpus is transformed, a lexical entry $w := C : I$ is extracted for each leaf that is aligned to one or more natural-language words. w is the sequence of aligned words, C is the CCG category of the leaf and I is the RCL interpretation of the leaf.

If more than one word is in w, we have a *multiword* lexical item. In this case w is just a list of word forms. If w contains only a single word, we append part-of-speech information to it to prevent lexical items from overgenerating. Parts of speech are obtained using the C&C POS tagger (Curran and Clark, 2003a), trained on the Penn Treebank.

Examples of lexical items are:

- $\langle$block/NN$\rangle$:= entity:$(entity{:}(type{:}(cube))$
- $\langle$cube/NN$\rangle$:= entity:$(entity{:}(type{:}(cube))$
- $\langle$on/IN$\rangle$:= spatial-relation/entity:
  $(spatial\text{-}relation : (relation : above))$
- $\langle$on, top, of$\rangle$:= spatial-relation/entity:
  $(spatial\text{-}relation : (relation : above))$

Note how different lexical entries with the same category and semantics account for lexical variation.

To allow for words that are not linked to the meaning representation, such as the two instances of *the* in Figure 4.3, we also consider a special type of lexical item which is semantically empty. The X here is a variable that can take on any category as value:

- $\langle w \rangle := (X/X){:}nil$
- $\langle w \rangle := (X\backslash X){:}nil$

4.3.3 Combinatory Rules

Although we have already produced and shown CCG derivation trees, we still have to specify the combinatory rules that license these derivations, given the lexical items we extracted. Our CCG primarily uses the standard rules of forward and backward application. However, we give a modified, non-standard and RCL-specific semantics to these rules. This makes notation simpler and ensures that the semantics of (most) intermediate constituents can themselves be (partial) RCL expressions rather than λ-terms with variables. This is important for interfacing with the spatial planner during parsing.

The two rules are defined as follows:

(2) *Forward application ($>^0$)*

a. For absorbing semantically empty words
$(X/X){:}nil \quad X{:}\mathbf{h} \quad \Rightarrow \quad X{:}\mathbf{h}$

b. For modifiers
$(X/X){:}\mathbf{a} \quad X{:}(t{:}\mathbf{h}) \quad \Rightarrow \quad X{:}(t{:}\mathbf{ah})$ where $\mathbf{a} \neq nil$

c. For non-modifiers
$(X/Y){:}(t{:}\mathbf{h}) \quad Y{:}\mathbf{c} \quad \Rightarrow \quad X{:}(t{:}\mathbf{hc})$ where $X \neq Y$

(3) *Backward application ($<^0$)*

a. For absorbing semantically empty words
$X{:}\mathbf{h} \quad (X\backslash X){:}nil \quad \Rightarrow \quad X{:}\mathbf{h}$

b. For modifiers
$X{:}(t{:}\mathbf{h}) \quad (X\backslash X){:}\mathbf{a} \quad \Rightarrow \quad X{:}(t{:}\mathbf{ha})$

c. For non-modifiers
$Y{:}\mathbf{c} \quad (X\backslash Y){:}(t{:}\mathbf{h}) \quad \Rightarrow \quad X{:}(t{:}\mathbf{ch})$ where $X \neq Y$

In words, the interpretations of modifiers and arguments are combined with those of the heads simply by adding the former as an additional child to the latter.

We also use a restricted form of forward composition to form chains of entity adjuncts:

(4) *Forward composition ($>^1$)*
$(\text{entity}/\text{entity}){:}\mathbf{a} \quad (\text{entity}/\text{entity}){:}\mathbf{b} \quad \Rightarrow \quad (\text{entity}/\text{entity}){:}\mathbf{ab}$

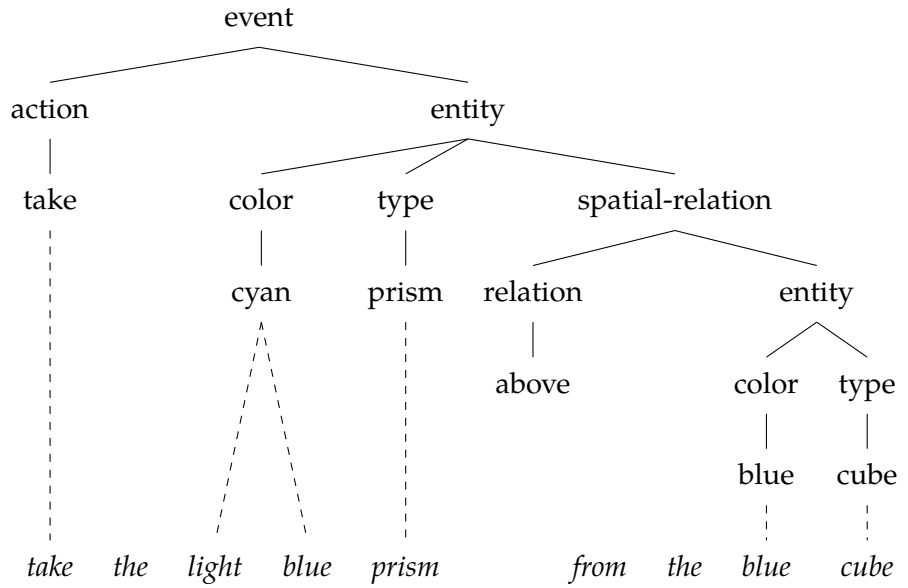


Figure 4.4: An RCL expression where one leaf is not linked to any word

This is motivated by our use of the spatial planner. Without forward composition, we would, e.g., not be able to build a constituent with the semantics $(entity{:}(color{:}green)(color{:}red)(type{:}cube\text{-}group))$ in the context of a stack consisting of green and red cubes, but no stack consisting exclusively of red cubes—the planner would filter out the intermediate constituent with semantics $(entity{:}(color{:}red)(type{:}cube\text{-}group))$.

Finally, we use type-changing rules. These are automatically extracted from the training data. Some of them account for unary productions within RCL expressions by introducing an additional internal node, such as the $destination$ node in Figure 4.2. For example:

(5) a. sp-relation:$\mathbf{h}$ $\Rightarrow$ destination:$(destination{:}\mathbf{h})$
b. entity:$\mathbf{h}$ $\Rightarrow$
spatial-relation/spatial-relation:$(measure{:}\mathbf{h})$

Others account for RCL leaves that are not linked to any words. A typ-

ical example is shown in Figure 4.4, where the annotation assumes a spatial modifying relation "above" between "prism" and "blue cube" that is however not linked to any word. Rules like the following deal with this by not only introducing an internal node, but also a branch leading to the unlinked leaf:

(6) a. $\text{entity}:\mathbf{h} \Rightarrow$
$\text{entity}/\text{entity}:(\textit{spatial-relation}:(\textit{relation}:\textit{above})\mathbf{h})$
b. $\text{entity}/\text{entity}:\mathbf{h} \Rightarrow$
$\text{entity}:(\textit{spatial-relation}:\mathbf{h}(\textit{type}:\textit{region}))$

4.3.4 Anaphora

Anaphora are marked in RCL *entity* expressions by the subexpression $(id:1)$ for antecedent entities and $(\textit{reference-id}:1)$ for anaphoric entities. The latter have the special type *reference*, in which case they are typically linked to the word *it*, or *type-reference*, in which case they are typically linked to the word *one*, as in *the yellow one*. More than one anaphoric relation in a command, and thus, other IDs than 1, are possible, but extremely rare. We do not explicitly try to resolve anaphora, but merely generate versions both with and without the *id* subexpression for each *entity* lexical item seen in training as an antecedent. We then rely on the parser and spatial planner to find a parse with the correct item marked as antecedent. If the spatial planner rejects a subexpression because it contains an unknown reference ID, we accept it anyway because the expression can later combine with another one that contains the antecedent. However, at the level of complete parses, those containing a *reference-id* expression but no *id* expression—or vice versa—are rejected. As a heuristic, we also reject parses where *reference-id* precedes *id* because we found this to be a noticeable source of errors, and no cataphora in the training data.

4.4 Training and Decoding

Following Zettlemoyer and Collins (2007), we use a CKY CCG parser in combination with simple perceptron updates: iterate over the training

corpus T times, for each sentence producing all parses. Each parse is characterized by a number of features and scored using a global weight vector. The weight vector is updated by subtracting the feature vector of the highest-scoring parse and adding the feature vector of the highest-scoring correct parse. No update is performed if the highest-scoring parse is correct, or no correct parse was found. We parallelize training using iterative parameter mixing (McDonald et al., 2010) with 12 shards.

4.4.1 Semantically Empty and Unknown Words

At the beginning of each parsing process, we consider each contiguous subsequence of words in the sentence and add all lexical items extracted from the training data $w := C{:}I$ to the chart where w matches the words in the subsequence. In the case of single words, we also require the POS tag in w to match the POS tag of the word in the sentence. We also add a forward and a backward semantically empty lexical item to the chart for each word. Finally, in decoding, the parser also has to deal with words not seen in training. For one, there are the *nil* items, so it is possible to treat the unknown words as semantically empty. In addition, for any unknown words, we look at other single-word lexical items with the same POS tag and generate lexical items with the same category and interpretation for the unknown word, hoping that features and the spatial planner will guide the parser to the right choice. To limit the search space, this is currently only done for nouns since we found the greatest lexical variation to occur with them.

4.4.2 Features

Each chart edge is characterized by the following local features:

- each lexical item $w := C{:}I$ used.
- each instance of a combinatory rule used, e.g., $>^0$.
- $\langle p, c, s \rangle$ for each lexical item used where p is the POS tag (or empty for multiwords). This allows to learn correlations between category/semantics pairs and particular parts of speech, primarily for unknown words.

- each instance of a type-changing rule used, together with the semantic head word of the constituent it roots, e.g., $\langle *_1, \text{in} \rangle$. This helps to learn not to use type-changing rules where they don't make sense. E.g. the word *squares* often heads entity descriptions that type-change into $measure$ phrases but the word *cube* doesn't.

- the root tag of the semantics of each constituent, together with the word to its immediate left, e.g., $\langle destination, \text{from} \rangle$. This example feature is indicative of typical erroneous parses where spatial adjuncts corresponding to *from*-phrases are misparsed as destination complements. The word *from* provides a strong clue against such a parse but would be ignored without such a feature because it is not aligned to any RCL node.

- the root tag of the semantics of each constituent, together with the first word in it, e.g., $\langle spatial\text{-}relation, \text{above} \rangle$.

4.4.3 The Spatial Planner

The spatial planner provided together with the data provides access to the context in which each command is to be interpreted. It can tell us for some RCL subexpressions, chiefly $entity$ expressions, whether they "make sense" given the context. For example, if the parser builds an edge with semantics $(entity : (type : cube)(color : red))$ but there is no red cube anywhere on the board, we can immediately reject the edge. Probably we picked $(entity : (type : cube))$ as the semantics for some unknown noun, like *prsim* (a misspelling), and now see we were wrong. Rejecting edges denoting absent entities assumes that no negations or hypothetical descriptions are used, which is the case for this task. By rejecting them, we avoid errors and reduce the search space.

The planner also helps resolve attachment ambiguities early: in the command *put the prism on the cube*, a constituent with semantics $(entity : (type : prism)(spatial\text{-}relation : (relation : above)(entity : (type : cube))))$ is a possible but incorrect parse. If we are lucky enough that no pyramid is actually sitting on a cube in the microworld, the planner will weed it out.

We have not yet explored making the fullest possible use of the spatial planner for checking the validity of $event$ or $sequence$ expressions, which would involve simulating changing the state of the world as a sequence of $event$ instructions is carried out. Currently we only filter out initial $event$ instructions with action $drop$ for scenes in which there is nothing initially in the robot's gripper to be dropped. RCL requires the action $move$ here instead, a distinction which is often not made in the natural language commands.

4.5 Experiments and Results

We trained the final model on the 2,500 training examples provided, although discarding those longer than 15 words to speed up training. In both training and decoding, we used a beam search strategy that only allows the highest-scoring 60 edges in each chart cell to be used for building further edges. To prevent an overuse of nil items while using positive evidence for nil items in the training data, the weights of all non-nil lexical items seen in the training data were initialized to 1, and those of all nil items corresponding to words seen unaligned in the training data to 0.5. All other feature weights were initialized to 0. The number of training epochs T was set to 3. These values were chosen experimentally using 80% of the training data, and another 10% of it as development test data.

Table 4.1 shows the results of the shared task. When using the spatial planner, our system parsed 789 of the 909 test sentences (86.80%) exactly correctly, making third place among the participating systems. When not using the spatial planner, it still parsed 720 sentences (79.21%) exactly correctly, making second place by a wide margin and proving to be one of the more robust systems in this scenario.

A preliminary analysis suggests most errors are related to pronoun ellipsis, the ambiguous word *one*, anaphora or attachment ambiguity. We believe some further careful feature engineering and extended use of the spatial planner could go a great length to improve accuracy further.

Table 4.1: Results of the shared task, adapted from Dukes (2014).

System	With planner	Without planner
Packard (2014)	92.50	90.50
Stoyanchev et al. (2014)	87.35	60.84
This work	86.80	79.21
Ljunglöf (2014)	86.10	51.50
Mattelaer et al. (2014)	71.29	57.76
Kate (2014)	-	45.98

4.6 Conclusions

The result shows that standard CCG-based semantic parsing techniques can be successfully applied to the domain of robotic spatial commands and profit from the integration of a spatial planner. We were able to apply CCG to the new domain and meaning representation formalism by sticking to CCG's basic principles, adapting the basic categories and interpretation rules, and exploiting composition to ensure smooth interfacing with the spatial planner. This illustrates CCG's generality and flexibility concerning linguistic domains, meaning representation formalisms and parsing strategies, including the incorporation of extra-linguistic cues. Seeing this, we feel justified in choosing CCG in moving on to bigger challenges in the next part, namely broad-coverage and cross-linguistic semantic parsing.

Part III

Broad-coverage Semantic Parsing

Chapter 5

Meaning Banking

5.1 Introduction

Every natural language processing task requires data—for evaluation, and for training. As we have seen in Chapters 3 and 4, for narrow-domain semantic parsing, a number of datasets exist. Most of these were relatively straightforward to build because only a limited, precisely circumscribed range of aspects of meaning is relevant to the respective task and has to be represented in the meaning representations. For broad-coverage semantic parsing, building datasets is considerably more difficult because a much larger range of potentially relevant aspects of meaning has to be considered, and annotated consistently.

The Groningen Meaning Bank (GMB; Basile et al., 2012b) is the first large-scale effort to build a broad-coverage corpus of texts paired with logical meaning representations covering the full propositional content (and projected content, such as presuppositions) of English text. The main emphasis is on *logical* meaning representations in the sense that they have a well-defined model-theoretic interpretation, in the vein of Montague (1970, 1973). We describe the meaning representation formalism in Section 5.2.

Another emphasis is on large size, so that the corpus can be used to study linguistic phenomena and their semantics in an empirical, data-driven way. The latest release of the GMB, version 2.2.0, contains 10,000

texts with a total of over a million tokens. At this scale, the complexity of the annotation task makes annotation from scratch by human annotators infeasible. Effective support of annotation by automatic tools is required. The GMB project opted for an approach dubbed "human-aided machine annotation", where a first annotation is done entirely automatically and then improved iteratively as more human annotation decisions become available. It also adopted the "release early, release often" principle from software engineering (Raymond, 1999) in order to enable early experimentation and create a tight feedback loop between the creators and the scientific community. This means that annotations are published even before they have been fully corrected by human annotators. They are then improved in subsequent releases. They are also accessible via the public Web interface of the GMB, where everybody can not only view but also edit the current development version.

Automatic annotation relies on a pipeline of tools that process natural language on different levels, from tokenization to tagging to parsing to semantic construction. To combine automatic and human annotation decisions in the most effective way, corrections should be made on the lowest possible level. For example, if a mistake by the part-of-speech tagger leads to a wrong analysis, it is preferable to correct the part-of-speech tag rather than the parse or the meaning representation directly. To make this possible, the GMB is opinionated about the relation between strings and their interpretation. It does not only contain texts and meaning representations, but also character-level annotation such as tokenization, token-level annotation such as tags and coreference chains and sentence-level annotation such as syntax trees. The annotation scheme for all these layers is described in Section 5.3. The process of building the GMB via human-aided machine annotation is described in Section 5.4.

In Section 5.5, we discuss the results produced by the GMB project, and briefly compare it to other recent semantic annotation efforts. Section 5.6 concludes.

5.2 Representing Meaning with Discourse Representation Structures

The meaning representation formalism of the GMB is rooted in Discourse Representation Theory (DRT). Since its original conception by Hans Kamp in the 1980s, DRT has been successfully used to model and study a wide variety of semantic phenomena, including anaphora, tense and aspect, plural semantics (Kamp, 1984; Kamp and Reyle, 1993), presuppositions (Van der Sandt, 1992), conventional implicatures (Venhuizen et al., 2014), rhetorical structure (Asher and Lascarides, 2003) and modality (Bos, 2004). DRT has thereby proven useful as a unifying theory for representing diverse aspects of meaning in a single type of representation. Moreover, these representations have well-defined model-theoretic interpretations (Kamp and Reyle, 1993) which can also be defined indirectly via a translation into first-order logic (Muskens, 1996; Bos, 2004), enabling the use of automated reasoning software for performing inference on natural-language text (Blackburn et al., 1999; Blackburn and Bos, 2005; Bos and Markert, 2006).

The meaning representations of DRT are called *discourse representation structures* (DRSs). A DRS is a pair of a list of *discourse referents* and a list of *conditions*. It is usually written as a box with an upper part for the discourse referents and a lower part for the conditions. The discourse referents are symbols referring to elements of the discourse, such as persons, things or events, and conditions are formulas asserting something about the referents. For example, a DRS representing the meaning of the sentence *Max is a famous singer* might look like this:

(1)

x1
named(x1, Max) famous(x1) singer(x1)

Here, one discourse referent is used: x1. Three conditions then assert that x1 is named Max, that x1 is famous, and that x1 is a singer, respectively.

In the following subsections, we describe in more detail the flavor

of DRT used in the GMB. For the purposes of this thesis, a slightly simplified presentation is sufficient. In particular:

- We ignore different types of name conditions, treating all names as being of the same type.
- We leave aside the treatment of tense and aspect, suppressing all related conditions from the annotation scheme.
- We deal only with sentences in isolation, not texts, and do not represent rhetorical structure.
- We do not deal with non-compositional phenomena such as anaphora, bridging or accommodation. We leave all anaphora unresolved in our meaning representations.

5.2.1 Basic Conditions

The basic conditions types used are

- one-place relation conditions of the form <symbol>(<ref>),
- two-place relation conditions of the form <symbol>(<ref>, <ref>),
- name conditions of the form named(<ref>, <string>),
- subset conditions of the form <ref>⊆<ref>,
- hybrid conditions of the form <ref>:<drs>,
- equality conditions of the form <ref>=<ref>,
- cardinality conditions of the form | <ref> | =<number>, and
- time conditions of the form timex(<ref>, <time expression>).

To represent proper names of individuals such as persons, locations or organizations, we use a name condition which relates a discourse referent to the name. For example: named(x1, Max).

Meanings of common nouns are represented using the sense inventory of WordNet 3.0 (Fellbaum, 1998). We use one-place relation conditions with canonical synset identifiers.[1] For example: book.n.02(x2).

Personal pronouns trigger conditions that describe them using the closest matching WordNet sense where applicable: person.n.01 for *I*, *me*, *you*, *we* and *us* and thing.n.12 for *they* (note that this is a supersense of person.n.01 but also of other senses, as *they* can refer to persons and non-persons). The third-person singular pronouns *he*, *him*, *she*, *her* and *it* are represented using the special (non-WordNet) predicate symbols male, female and neuter.

We use a neo-Davidsonian event semantics. A verb contributes its own discourse referent, representing the *event* the verb describes. WordNet verb sense identifiers are used as one-place predicate symbols over events, e.g., pass.v.05(e3). The participants in the event are related to it using two-place relation conditions using symbols corresponding to semantic roles. For participants central to the event, we use the most specific applicable semantic role from the unified VerbNet/LIRICS semantic role hierarchy defined by Bonial et al. (2011). For example: Recipient(e3, x1).

The DRS representing the meaning of the sentence *Mary gave me a book* then looks like this:

(2)

e1 x2 x3 x4
pass.v.05(e1)
named(x2, Mary)
Agent(e1, x2)
person.n.01(x3)
Recipient(e1, x3)
book.n.02(x4)
Theme(e1, x4)

Note that the names of discourse referents are arbitrary. By convention, we use letter-number combinations where the number is unique within

[1] Each synset is represented by the first lemma in it followed by its part of speech (n, v, a, s or r) and the corresponding sense number. This is the convention used, e.g., in NLTK (Bird et al., 2009).

the DRS and the letter indicates the type of the discourse referent: e for events, s for manners, p for propositions and x for others.

The representation does not distinguish between the meanings of singular and plural nouns. However, coordinated noun phrases introduce a discourse referent for each conjunct and an additional discourse referent to represent the entire group, related through subset conditions. For example, for the sentence *Cats and dogs are singing*:

(3)

x1 x2 x3 e4
cat.n.01(x1)
x1 ⊆ x3
dog.n.01(x2)
x2 ⊆ x3
sing.v.02(e4)
Agent(e4, x3)

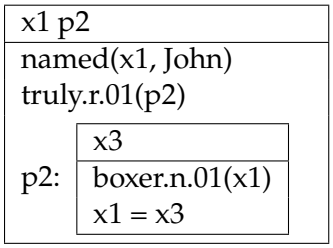
A DRS can be embedded within another DRS via a so-called hybrid condition. A hybrid condition names the inner DRS with a discourse referent that can then also be referred to by other conditions in the outer DRS. This is, for example, used to represent clausal complements as in *John likes to swim*:

(4)

x1 e2 p3	
named(x1, John)	
like.v.02(e2)	
Experiencer(e2, x1)	
Stimulus(e2, p3)	
p3:	e4
	swim.v.01(e4)
	Theme(e4, x1)

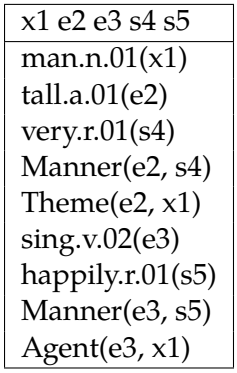
Hybrid conditions are also used, in combination with equality conditions, to represent the meaning of copulas connecting two noun phrases. The hybrid condition allows for a reified representation and thereby for modification, as in *John is really a boxer*:

(5)



Like verbs, the meanings of adjectives and adverbs are represented by introducing an additional discourse referent which is related to the discourse referent that the adjective/adverb describes, by means of a semantic role. Adjectives introduce events that use the Theme role to describe things. Adverbs describing manners, times and places introduce discourse referents that themselves fill the corresponding VerbNet/LIRICS role (Manner, Time or Place) of the event they further describe. The reified representation of adjectives and adverbs allows for straightforward representation of additional modifiers, e.g., intensifiers such as *very*. For example, the sentence *A very tall man is singing happily* is represented as follows:

(6)



Occasionally, adjectives have additional arguments, in which case we also use VerbNet/LIRICS roles, For example, in *He is not like us*, both *he* and *us* fill Theme roles of *like*, and in *My house is not far from here*, *My house* fills the Theme role of *far*, and *here* fills the Source role.

An exception to the above are fully intensional adverbs (cf. Larson, 2002, Section 3), which we represent as not introducing a discourse referent but an embedded DRS as in *John allegedly sang*:

(7)

x1 p2
named(x1, John)
p2: e3 | sing.v.02(e3) Agent(e3, x1)
allegedly.r.01(p2)

The meaning of adjectives in the comparative degree is not explicitly distinguished. However, if there is an object of comparison, it is related to the event using the special *than* relation. For example, for the sentence *Jane is taller than John*:

(8)

x1 e2 x3
named(x1, Jane)
tall.a.01(e2)
Theme(e2, x1)
than(e2, x3)
named(x3, John)

For relations other than verb and adjective arguments, which are covered by VerbNet/LIRICS roles, a special inventory of two-place relation symbols is used, based on the set of English prepositions and subordinating conjunctions. They are used to represent the meaning of:

- prepositional adjuncts to nouns, using the respective preposition,
- prepositional adjuncts to verbs, using the respective preposition,
- possessives, using *of*,
- noun-noun compounds, using the preposition semantically best describing the relation between the parts,

- bare-noun-phrase adjuncts referring to times, using *at* for clock-times and other points in time, *on* for days and *in* for other periods of time,
- bare-noun-phrase adjuncts expressing per-unit amounts as in *$ 150 a barrel*, using the preposition semantically best describing the relation, usually *for*,
- adverbial clauses, using the respective conjunction.

For example, the sentence *Mary's friend hosted a beach party with dancing Friday for John because he graduated* receives this representation:

(9)

x1 x2 e3 x4 x5 x6 x7 x8 p9

named(x1, Mary)
friend.n.01(x2)
of(x2, x1)
host.v.01(e3)
Agent(e3, x2)
beach.n.01(x4)
party.n.02(x5)
on(x5, x4)
dancing.n.01(x6)
with(e3, x6)
for(e3, x5)
named(x7, Friday)
on(e3, x7)
named(x8, John)
for(e3, x8)
because(e3, p9)

p9: e10 | graduate.v.01(e10) male(x8) Agent(e10, x8)

Cardinalities are represented using cardinality conditions. For example, for the sentence *Two dogs barked*, we have:

(10)

x1 e2
dog.n.01(x1)
$\|x1\| = 2$
bark.v.04(e2)
Agent(e2, x1)

Time periods are represented as named entities in the case of week days, otherwise as entities related using time conditions to a string of the form SYYYYMMDDHHmm where S is a sign ($-$ for B.C., $+$ for A.D.), YYYY is a year, MM is the month of the year, DD is the day of the month, HH is the 24-hour-clock hour of the day, mm is the minute of the hour. The places for date and time parts not expressed are filled up with X's. For example, the string for *in the 1980's* is +198XXXXXXXX, for *250 B.C.*, -250XXXXXXXX, for *July*, XXXXX07XXXXXX, for *8:30 P.M.*, XXXXXXXX2030, etc. This is the DRS for the sentence *The lecture starts at 8 o'clock*:

(11)

x1 e2 x3
lecture(x1)
start(e2)
at(e2, x3)
timex(x3, XXXXXXXX0800)

5.2.2 Complex Conditions

Multiple conditions listed in the same DRS implicitly form a logical conjunction. For example, we can read the following DRS logically as *There are entities x1 and e2 such that x1 is a policeman* ***and*** *e2 is a singing event* ***and*** *x1 is the Agent of e2*:

(12)

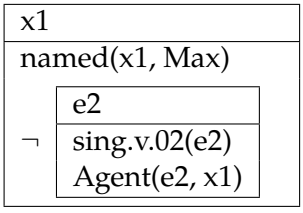
x1 e2
policeman.n.01(x1)
sing.v.02(e2)
Agent(e2, x1)

Logical relationships between conditions other than conjunction can be expressed by wrapping the respective conditions and discourse referents in embedded DRSs under a logical operator. A logical operator and its DRS argument(s) form a *complex condition*. The complex condition types are

- negation conditions of the form ¬ <drs>,
- implication conditions of the form <drs>⇒<drs>,
- disjunction conditions of the form <drs> ∨ <drs>,
- question conditions of the form <drs> ? <drs>,
- necessity conditions of the form □ <drs>, and
- possibility conditions of the form ◇ <drs>.

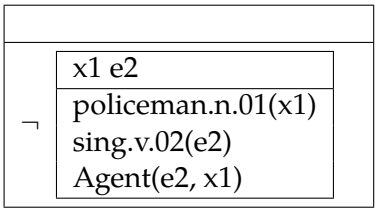
Negation conditions are used to represent the meaning of a simple negation in a sentence, as in *Max does not sing*:

(13)



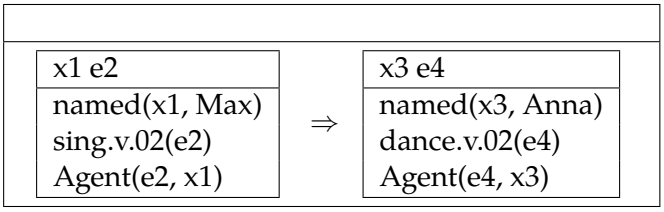
They are also used to represent negative existential quantification, as in *No policeman sings*:

(14)



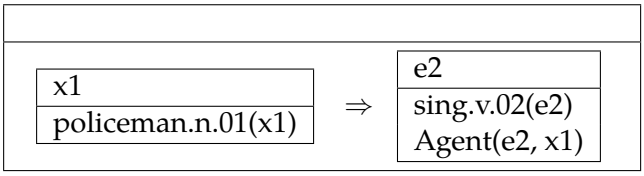
An implication condition asserts that the first argument implies the second argument. Implication conditions are used to express the meaning of simple conditionals, as in *If Max sings, Anna dances*:

(15)



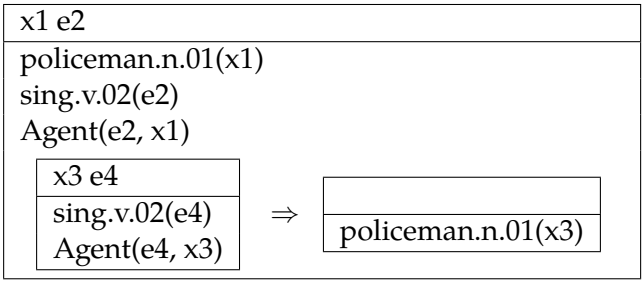
They are also used to represent universal quantification, as in *Every policeman sings*:

(16)



They are furthermore used to represent the meaning of the focus particle *only*, as in *Only policemen sing*. The representation asserts the truth of the prejacent *Policemen sing* (cf. Horn, 1996), then adds an implication with the unfocused material *sing* represented in the antecedent and the focused material *policemen* represented in the consequent:

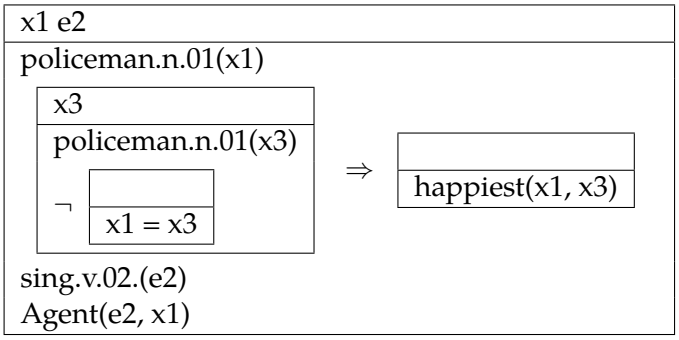
(17)



Another important trigger for implication conditions are superlatives. An NP of the form *the Yest Z* is represented by asserting that a "Yer-than-relation" holds between the discourse referent it introduces and every

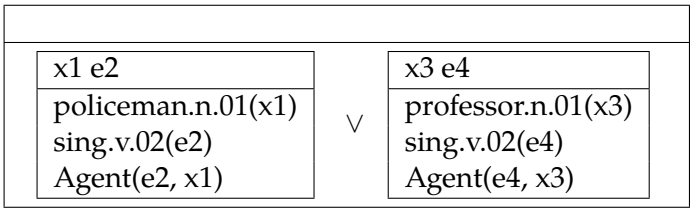
other Z. For example, for the sentence *The happiest policeman sings*:

(18)



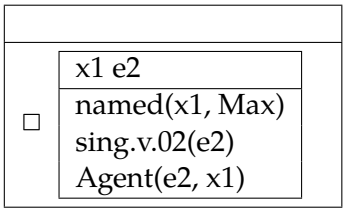
Disjunction conditions are used to represent alternatives as in *A policeman or a professor sings*:

(19)

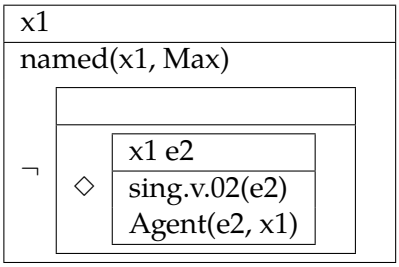


Necessity and possibility conditions are usually triggered by modal verbs as for example in *Max must sing* or *Max cannot sing*, respectively:

(20) a.

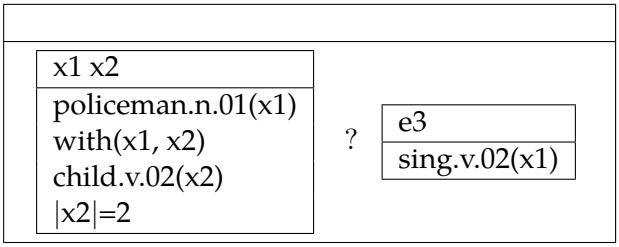


b.



A question condition is used to represent the meaning of (direct or indirect) *wh*-questions. Its first argument represents the restrictor of the question, and its second, the nuclear scope. For example, for the question *which policeman with two children sings*:

(21)



The meaning of yes/no questions is not distinguished in our representation from that of the corresponding declarative sentences.

5.2.3 Projection Pointers

Our meaning representation language as presented so far lacks a pragmatically and semantically important distinction: that between *asserted* (or *at-issue*) content and that between *projected* (or *not-at-issue*) content (Simons et al., 2010; Tonhauser et al., 2013). Consider the following five sentences:

(22) a. A policeman sings.
b. The policeman sings.
c. He sings.
d. John's sister sings.
e. John, who has a sister, sings.

The main propositional content of each sentence is that somebody sings. This is *asserted* content. In (22-a), asserted content is all there is: there is a policeman, there is a singing event, and the former is the agent of the latter. By contrast, in (22-b)–(22-e), the underlined part contributes *projected* content. All projected content is in some sense "background" or "extra" information that is not what the sentence is "about" but which nonetheless supports certain inferences we can draw from the sentence. In (22-b), the definite article carries a *presupposition* that a policeman exists and is uniquely identifiable in the discourse context. In (22-c), the personal pronoun similarly requires that there is some male person uniquely identifiable in context, to which it then refers. In (22-d), the possessive construction contributes the presupposition that John has a sister. In (22-e), the non-restrictive relative clause contributes a *conventional implicature* supporting the same inference.

The common defining characteristic of all projected content is that it can survive unchanged when the whole utterance is embedded under an entailment-canceling operator like negation, a conditional, a modal or a question. Consider the negations of the above sentences:

(23) a. It is not the case that a policeman sings.
b. It is not the case that the policeman sings.
c. It is not the case that he sings.
d. It is not the case that John's sister sings.
e. It is not the case that John, who has a sister, sings.

While none of (23-a)–(23-e) support the same *asserted* inferences as examples (22-a)–(22-e), the *projected* inferences of (23-b)–(23-e) are still supported by the most natural reading of (23-b)–(23-e): (23-b) still carries the presupposition that a uniquely identifiable policeman exists, (23-c) still requires a uniquely identifiable male referent, (23-d) and (23-e) still support the inferences that John has a sister. The entailments of projected content are "projected" out of the local scope and therefore survive the entailment-canceling operator.

To distinguish asserted from projected content, the GMB uses an extension of DRT called Projective Discourse Representation Theory (PDRT; Venhuizen et al., 2013b). Each DRS in PDRT (PDRS) carries an

arbitrary but unique *label* which by convention is written as a number, and every discourse referent and condition carries a *projection pointer*, which may or may not be the same as the label of a containing DRS. Referents and conditions representing asserted content have the same label as the DRS in which they occur. Those representing projected content have a different label, so as to indicate that they should be interpreted in a different context. For example, (22-a)–(22-e) are represented as follows:

(24) a. A policeman sings

1

1 ← x1 1 ← e2
1 ← policeman.n.01(x1)
1 ← sing.v.02(e2)
1 ← Agent(e2, x1)

b. The policeman sings

2

1 ← x1 2 ← e2
1 ← policeman.n.01(x1)
2 ← sing.v.02(e2)
2 ← Agent(e2, x1)

c. He sings

2

1 ← x1 2 ← e2
1 ← male.a.01(x1)
2 ← sing.v.02(e2)
2 ← Agent(e2, x1)

d. John's sister sings

3

1 ← x1 2 ← x2 3 ← e3
1 ← named(x1, John)
2 ← sister.n.01(x2)
2 ← of(x2, x1)
3 ← sing.v.02(e3)
3 ← Agent(e3, x2)

e. John, who has a sister, sings

3

1 ← x1 2 ← x2 2 ← e3 3 ← e4
1 ← named(x1, John)
2 ← sister.n.01(x2)
2 ← have.v.01(e3)
2 ← Pivot(e3, x1)
2 ← Theme(e3, x2)
3 ← sing.v.02(e4)
3 ← Agent(e4, x1)

5.2.4 The Syntax-semantics Interface

The GMB includes not only DRS annotations for English sentences but also CCG derivations that relate sentences to the DRSs. That is, a compositional analysis is given where each word has a category and an interpretation (cf. Chapter 2). Interpretations are λ-PDRSs (Bos, 2009; Venhuizen et al., 2013b). These are expressions that when combined via CCG's combinatory rules eventually combine into full PDRSs representing the meanings of full sentences or texts. What the possible λ-PDRSs for each word are depends, among other things, on its category. In this section we present the most important kinds of interpretations for the most important categories.

The interpretation of a **noun** (N) is a *property*, more specifically, a function that takes a discourse referent and returns a PDRS with one condition. For example, here is a possible interpretation for the word *book*:

(25) $\lambda x.$ 1 | | / 1 ← book.n.01(x)

The interpretation of an **attributive adjective** (N/N) is a function from properties to properties. For example, intersective attributive adjectives apply the property that is the interpretation of the noun they are modifying to a variable standing for the eventual discourse referent and merge the resulting DRS with another DRS which contains additional conditions contributing the adjective meaning. A possible interpretation for *big* is thus:

(26) $\lambda p.(\lambda x.($ 1 | 1 ← e / 1 ← big.a.01(e), 1 ← Theme(e, x) $+ (p@x)))$

The $+$ operator denotes an *assertive merge* (Venhuizen et al., 2013b): the inputs are two PDRSs. The output is one PDRS with the union of the discourse referents and the union of the conditions of both inputs. The label of the output PDRS is taken from the second input PDRS and is also unified with the label of the first, so that assertive content stays assertive. For example:

(27) 1 | 1 ← e / 1 ← big.a.01(e), 1 ← Theme(e, x) + 2 | 2 ← x / 2 ← book.n.01(x) = 2 | 2 ← e 2 ← x / 2 ← big.a.01(e), 2 ← Theme(e, x), 2 ← book.n.01(x)

The interpretation of a **noun phrase** (NP) is a generalized quantifier, i.e., a function from properties to PDRSs. For example, the NP *someone* takes a property, applies it to a discourse referent—allowing the property to make predications about it—and finally merges the resulting PDRS with a PDRS in which the discourse referent is introduced and established as a person:

(28) $\lambda p.($ $\begin{array}{|l|}\hline 1 \\ \hline 1 \leftarrow x \\ \hline 1 \leftarrow \text{person.n.01}(x) \\ \hline \end{array}$ $+ (p@x))$

Determiners (NP/N) turn nouns (N) into noun phrases (NP), so their interpretations are functions from properties to generalized quantifiers. Since generalized quantifiers are functions from properties to PDRSs, a determiner interpretation can also be seen as a function that takes two properties p and q and returns a PDRS that relates them in some way. For example, the indefinite determiner *a/an* just introduces a discourse referent, applies both properties to it and merges the resulting PDRSs:

(29) $\lambda p.(\lambda q.($ $\begin{array}{|l|}\hline 1 \\ \hline 1 \leftarrow x \\ \hline \\ \hline \end{array}$ $+ ((p@x) + (q@x))))$

By contrast, the universal quantifier *every/each/all/any* introduces an implication condition where p serves as antecedent and q as consequent:

(30) $\lambda p.(\lambda q.$ $\begin{array}{|l|}\hline \\ \hline 1 \leftarrow (\begin{array}{|l|}\hline 2 \\ \hline 2 \leftarrow x \\ \hline \\ \hline \end{array} + (p@x)) \Rightarrow (q@x) \\ \hline \end{array}$ $)$

Words that trigger presuppositions, anaphoric relations or conventional implicatures use a different merge operation: *projective merge* ($*$). Projective merge is defined like assertive merge, except that labels are not unified. Thereby, the contents of the first input PDRS become projected in the output PDRS. For example:

(31) $\begin{array}{|l|}\hline 1 \\ \hline 1 \leftarrow x \\ \hline 1 \leftarrow \text{policeman.n.01}(x) \\ \hline \end{array}$ $*$ $\begin{array}{|l|}\hline 2 \\ \hline 2 \leftarrow \text{e} \\ \hline 2 \leftarrow \text{sing.v.02}(e) \\ 2 \leftarrow \text{Agent}(e, x) \\ \hline \end{array}$ $=$ $\begin{array}{|l|}\hline 2 \\ \hline 1 \leftarrow x \\ \hline 1 \leftarrow \text{policeman.n.01}(x) \\ 2 \leftarrow \text{sing.v.02}(e) \\ 2 \leftarrow \text{Agent}(e, x) \\ \hline \end{array}$

One example of a lexical interpretation using projective merge is that of

the definite article *the*:

(32) $\lambda p.(\lambda q.((\ \overset{1}{\boxed{\begin{array}{|l}\hline 1 \leftarrow x \\ \hline \\ \hline\end{array}}}\ + (p@x)) * (q@x)))$

Another is that of the pronoun *she*:

(33) $\lambda p.(\ \overset{1}{\boxed{\begin{array}{|l}\hline 1 \leftarrow x \\ \hline 1 \leftarrow \text{female}(x) \\ \hline\end{array}}}\ * (p@x))$

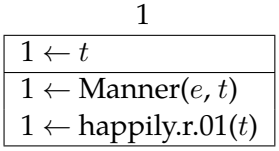
A third is that of the possessive suffix *'s*:

(34) $\lambda o.(\lambda p.(\lambda q.(o@\lambda y.((\ \overset{1}{\boxed{\begin{array}{|l}\hline 1 \leftarrow x \\ \hline 1 \leftarrow \text{of}(x, y) \\ \hline\end{array}}}\ + (p@x)) * (q@x)))))$

The interpretation of a **sentence** ($S[X]$ for some feature X) might be expected to just be a PDRS. In fact, we represent it as a function from properties to PDRSs. This way, they can be applied to an additional property m to modify the event at the core of the sentence interpretation. For example, here is the interpretation of the sentence *A policeman sings*:

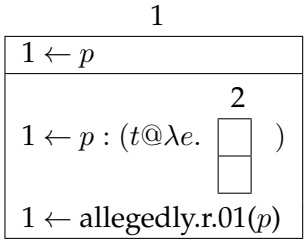
(35) $\lambda m.(\ \overset{1}{\boxed{\begin{array}{|l}\hline 1 \leftarrow x \quad 1 \leftarrow e \\ \hline 1 \leftarrow \text{policeman.n.01}(x) \\ 1 \leftarrow \text{sing.v.02}(e) \\ 1 \leftarrow \text{Agent(e, x)} \\ \hline\end{array}}}\ + (m@e))$

Sentence modifiers ($S\backslash S$ or S/S) which semantically modify the event exploit this mechanism by applying the sentence interpretation t to a property contributing the appropriate conditions—and return yet another function from properties to PDRSs, for any further modifiers. For example, here is a possible interpretation of the adverb *happily*:

(36) $\lambda t.(\lambda m.(t@(\lambda e.$ 

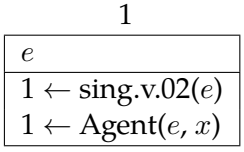
$+ (m@e))))$

Sentence modifiers that semantically modify the proposition expressed in the sentence, e.g., intensional adverbs like *allegedly*, apply the sentence interpretation t to an empty property instead and embed the resulting PDRS:

(37) $\lambda t.(\lambda m.($ 

$+ (m@p)))$

Verb phrases ($S[X]\backslash NP$ for some feature X) are "sentences missing a subject", thus their interpretations are functions from NP interpretations to sentence interpretations. Recall that NP interpretations are functions from properties to DRSs. A VP interpretation thus applies the subject NP interpretation s to a property, *viz.* one that contributes the event, the semantic roles and any objects inside the VP. For example, here is a possible interpretation of the VP *sings*:

(38) $\lambda s.(\lambda m.(s@\lambda x.($ 

$+ (m@e))))$

Predicative adjectives ($S[\mathrm{adj}]\backslash NP$) are analyzed like verb phrases. For example, here is a possible interpretation of *successful*:

(39) $\lambda s.(\lambda m.(s@\lambda x.($ $\overset{1}{\boxed{\begin{array}{l} e \\ \hline 1 \leftarrow \text{successful.a.01}(e) \\ 1 \leftarrow \text{Theme}(e,\, x) \end{array}}}$ $+ (m@e))))$

The interpretation of the **copula** ($(S[X]\backslash NP)/(S[\text{adj}]\backslash NP)$ for some feature X) connecting subjects to predicative adjectives is just a wrapper around that of the adjective. The same is true of **auxiliary verbs** like the perfect *have* ($(S[X]\backslash NP)/(S[\text{pt}])$ for some feature X) since we ignore tense:

(40) $\lambda v.(\lambda s.(\lambda m.((v@s)@m)))$

VP modifiers ($(S\backslash NP)\backslash(S\backslash NP)$) are very similar to sentence modifiers, they only need to "delay" the contribution of their content until after combination with the subject. For example, here is the semantics of the VP modifier *in a tent*:

(41) $\lambda v.(\lambda s.(\lambda m.((v@s)@\lambda e.($ $\overset{1}{\boxed{\begin{array}{l} 1 \leftarrow x \\ \hline 1 \leftarrow \text{tent.n.01}(x) \\ 1 \leftarrow \text{in}(e,\, x) \end{array}}}$ $+ (m@e)))))$

Figure 5.1 illustrates how the λ-DRSs and the combinatory rules interact to produce sentence interpretations from word interpretations.

5.3 Token-level Annotation

Once a sentence is assigned its correct CCG derivation, and each word is assigned its correct interpretation, the interpretation of the sentence follows deterministically. Finding the correct CCG derivation depends largely on assigning each token (word or punctuation mark) the correct category, also known as supertagging or "almost parsing" (Srinivas and Joshi, 1999; Clark and Curran, 2007). Almost all annotation decisions can thus be made at the token level, and this is what the annotation methodology of the GMB focuses on.

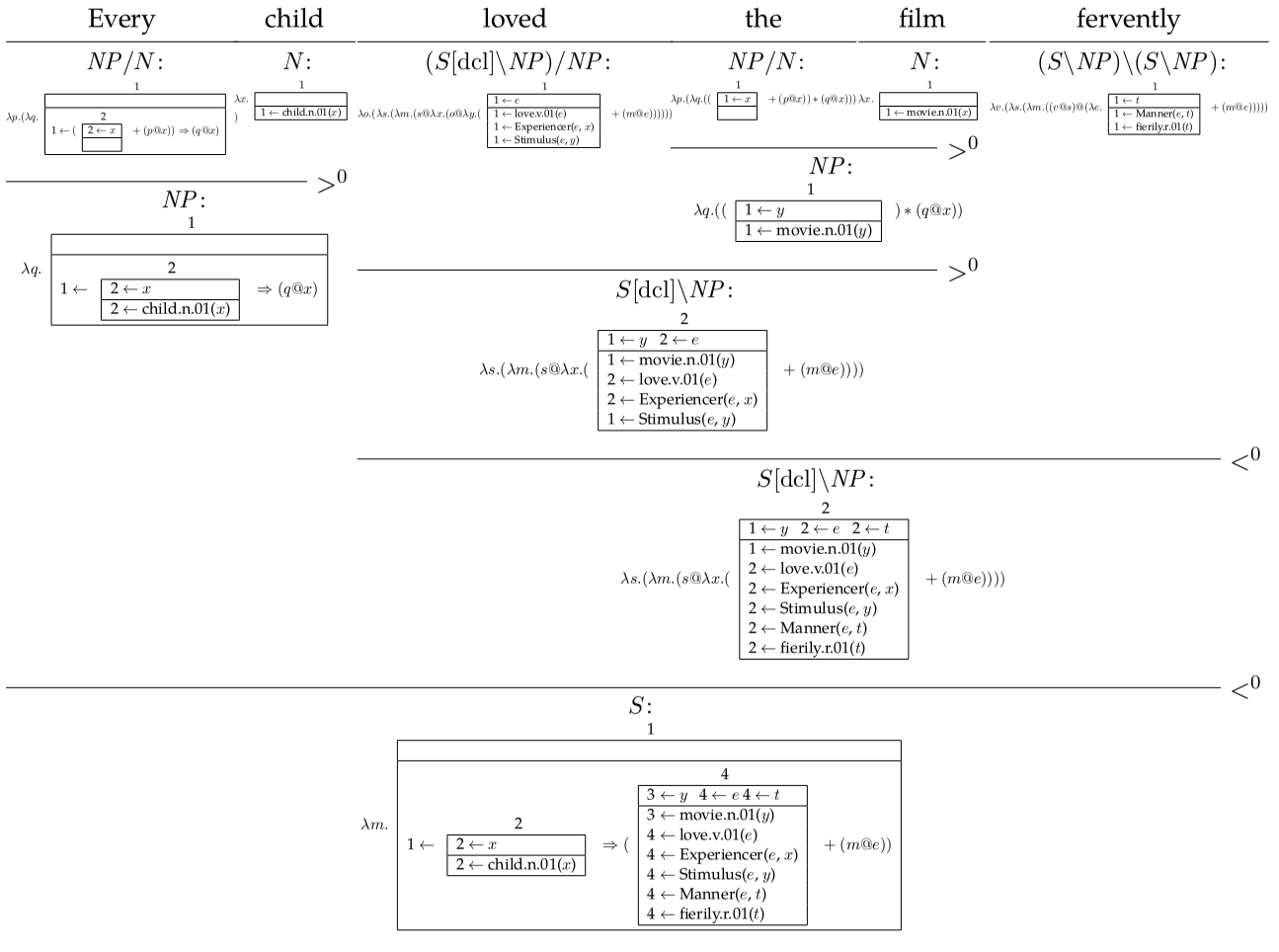


Figure 5.1: Example derivation with λ-DRS interpretations.

Before there are any tokens to annotate, they must first be identified in the raw documents, which come as sequences of characters. Furthermore, the boundaries of sentences must be determined before parsing can begin. This is done in segmentation, which we describe in Section 5.3.1.

In choosing the correct interpretation for a token, there are many variables to consider. As we have seen in the previous section, the λ-DRS of content words depends, for example, on their senses and the semantic roles they assign to their arguments. As we will see, they can also depend on the scope orders of their arguments. The interpretations of function words do not follow from categories alone either, e.g., definite and indefinite articles have different interpretations; so do personal pronouns, depending on their gender; and so on.

To make the multitude of variables to consider more manageable both for human annotators and for the various natural-language processing tools supporting them, we designed a tagging scheme with a total of ten annotation layers, each of which can be annotated independently of the others. Many of the layers correspond to established tagging tasks, like part-of-speech tagging, supertagging or word-sense disambiguation. The tagging scheme was designed so that it contains all the information required to choose the correct λ-DRS automatically. It was also designed so that the correct λ-DRS for a token follows from its tags alone, without looking at context. Where context is required for making decisions, this has to be taken into account during tagging. We describe the tagging scheme in Section 5.3.2.

As an example of how token-level tagging can be used to annotate complex semantic phenomena involving several tokens, we have a closer look at the *scope* tagging layer in Section 5.3.3.

5.3.1 Segments

Segmentation of text into tokens and sentences is widely regarded as a solved problem. For languages written with spaces between words, rule-based systems solve it with near-perfect accuracy for many applications. However, not all aspects of the task are trivial. Most notably, periods have to be disambiguated based on context—they can termi-

```
I don't think New York is as supercali-
SOTITIIOTIIIIOTIIIIIIIOTIOTIOTIIIIIIIOO
fragilisticexpialidocious as it used to
IIIIIIIIIIIIIIIIIIIIIIIOTIOTIOTIIIOTIO
be. But I still want to go there.
TITOSIIOTOTIIIIOTIIIOTIOTIOTIIIITO
```

Figure 5.2: Example of IOB-labeled characters, with two kinds of B-tags: S for the beginning of a sentence, and T for the beginning of a token.

nate a sentence, but they can also just be part of an acronym. Similarly, if units such as *New York* or *in spite of* are to be treated as a single token, considerable knowledge is required of the tokenizer. Rule-based systems can only acquire such knowledge through careful writing and testing of rules for each new language and domain (Silla Jr. and Kaestner, 2004). In the data-driven spirit of the GMB, a system was desired that allows annotators to correct *instances* of mistakes in the data, and is able to learn from them to treat other instances in the same way.

We approach the task as a character tagging problem following a variant of the IOB scheme which is widely used, e.g., in chunking. The first character in a token is assigned the `S` tag if it also starts a sentence, otherwise `T`. Other characters that are part of tokens are tagged `I`. Characters that are not part of a token, such as whitespace, are tagged `O` (Evang et al., 2013). An example of a text tagged in this way is shown in Figure 5.2. Compared to the more traditional approach of tokenizing and sentence-segmenting a text by inserting/removing whitespace and newlines, this one has a couple of advantages: it makes it unnecessary to edit the text itself, ensuring that the original raw text is preserved. It makes it straightforward to apply sequence tagging techniques to the task. It allows for representing tokens that contain whitespace, such as *New York*. Tokens that in the raw file are interrupted by artifacts such as hyphenation, like *supercalifragilisticexpialidocious*, can be represented by tagging the interrupting characters as `O` and then resume the token with `I`.

We largely follow the segmentation scheme of the Penn Treebank

token	Mary	likes	her	lunch	box	.
lexical category						
part of speech	NNP	VBZ	PRP$	NN	NN	.
lemma	mary	like	her	lunch	box	.
category	N	$(S[\text{dcl}]\backslash NP)/NP$	NP/N	N/N	N	$S\backslash S$
lexical meaning						
named entity type	PER					
animacy	HUM		HUM	CNC	CNC	
word sense		like.v.01		lunch.n.01	box.n.01	
contextual meaning						
thematic roles		[Stimulus, Experiencer]				
other relations			of	for		
scope		neutral				
co-reference			0,4			

Table 5.1: An example sentence annotated on all tagging layers.

(Marcus et al., 1993), characterized by treating punctuation marks (except word-internal hyphens) as independent tokens and splitting off the possessive suffix *'s* and the *n't* in contractions such as *won't* as separate tokens. We depart from the Penn Treebank scheme in annotating names that have spaces in them, such as *New York*, as a single token.

5.3.2 Tags

Every token is annotated on a total of ten tagging layers. If a tagging layer is not applicable to a token, the respective tag is the empty string. An example of a sentence tagged on all ten layers is given in Table 5.1.

Lexical Category Tokens are annotated for **part of speech** following the tagging scheme of the Penn Treebank (Marcus et al., 1993) and for **CCG category** following CCGbank (Hockenmaier and Steedman, 2007). The **lemma** tagging layer contains the uninflected, lower-case forms of each token.

Named Entity Type Proper nouns and demonyms are annotated for the type of named entity they refer to: one of PER for person, GEO for location, ORG for organization, TIM for times, (e.g., names of months or weekdays), EVE for events (such as Katrina, the hurricane), ART for

artifacts (e.g., names of products such as iPhone), NAT for natural phenomena (such as H5N1, the virus) and GPE (geo-political entity) for names of, e.g., countries where the name does not clearly refer to only one of the location or the political organization. The tagging scheme is loosely based on that of Sekine et al. (2002) and described in more detail in Bos et al. (2017).

Animacy Nouns and pronouns are annotated with tags that indicate whether they refer to animate or inanimate entities. Following the annotation scheme of Zaenen et al. (2004), animate entities are further subdivided into the classes HUM (Human), ORG (Organization), ANI (Animal), MAC (Machine) and VEH (Vehicle), and inanimate entities into LOC (Place), NCN (Non-concrete), CNC (Concrete) and TIM (Time).

Word Sense To resolve word-level homography and polysemy, common nouns, verbs, adjectives and adverbs are annotated with a WordNet synset identifier.

Thematic Roles Thematic roles specify the semantic relation of a verb (adjective, adverb) to its argument. We annotate them lexically following the method proposed by Bos et al. (2012): the head of the respective construction (i.e., the verb, adjective or adverb) is annotated with a list of roles, each indicating the role associated with the corresponding argument slot. For example, the verb *likes* has category $(S\backslash NP)/NP$. Its first argument slot is for the object NP (on the right), the second, for the subject NP (on the left). The list of roles [Stimulus, Experiencer] it is annotated with assigns the two arguments the roles Stimulus and Experiencer, respectively.

Other Relations The annotation of other relations between a functor and an argument similarly works by annotating the respective relation on the functor, with the difference that there is only ever one relation per functor in this case and therefore no list is needed. In our example, a *lunch box* is a *box for lunch*, so the noun modifier *lunch* is annotated with

the *for* relation, and the determiner *her* introduces an *of* (possessive) relation.

Scope Similarly, scope, a relation between noun phrases, is not annotated on the noun phrases themselves, but on the functors (like verbs and prepositions) that take those noun phrases or phrases containing them as arguments. We will have a closer look at this in the following subsection.

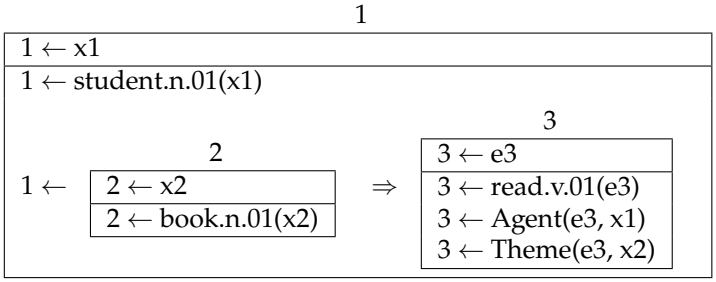
Coreference The final tagging layer is for coreference. In our semantic construction framework, coreference is not resolved during compositional construction of DRSs, but later as a post-processing step. The coreference annotation of tokens thus does not affect the choice of lexical interpretation for it. It purely serves as information for the post-processing step. We annotate coreference on referring elements such as pronouns by indicating the character offsets of the head word of a noun phrase in the same coreference chain. In our example, the possessive determiner *her* refers to *Mary*, a token that here is mentioned at the very beginning of the document (start offset 0) and is four characters long (therefore, end offset 4).

5.3.3 Quantifier Scope

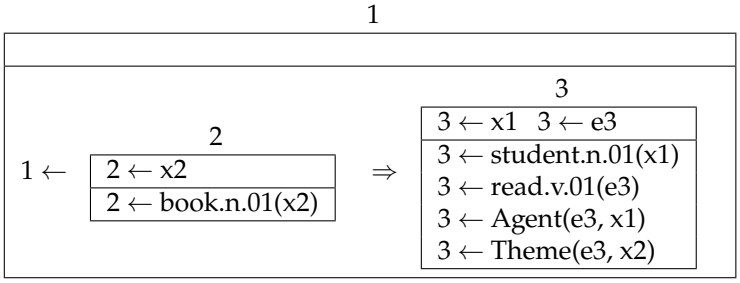
Quantifier scope is among the most intensely studied phenomena in formal semantics. Let us start with the observation that the sentence *Some student read every book* has two distinct readings which do not seem to stem from syntactic ambiguity (*some student* is the subject, *read* is the verb and *every book* is the object in any case). The two readings, given as PDRSs, are:

(42) Some student read every book.

a. Subject wide scope



b. Object wide scope



In the first case, there is a single student who read every book, in the second; every book was possibly read by a different student. In general, with universally quantified noun phrases such as *every student*, the question arises which of the discourse referents introduced by other noun phrases should be introduced outside of the implication condition, as x1 in (42-a), and which should be introduced inside its consequent, as x1 in (42-b). This is the problem of *quantifier scope disambiguation* (Higgins and Sadock, 2003; Andrew and MacCartney, 2004; Srinivasan and Yates, 2009; Manshadi and Allen, 2011). Quantifier scope can be formalized as an ordering relation between all noun phrases in a sentence, where noun phrases with a higher rank outscope (i.e., introduce their discourse referents at a higher position in the hierarchy of nested DRSs than) noun phrases with a lower rank.

In research on semantic construction, many efforts have been directed towards *avoiding* quantifier scope disambiguation, or at least at keeping it out of the process of compositional semantic construction. These efforts employ the strategy of *quantifier scope underspecification*,

i.e., they construct underspecified semantic representations from which fully specified semantic representations for all scope orders can be constructed later, if needed. Frameworks and formalisms for underspecification are presented for example in (Reyle, 1993; Bos, 1996; Egg et al., 2001; Copestake et al., 2005). These approaches essentially treat quantifier scope as a non-compositional phenomenon that needs to be dealt with at a higher level.

As Manshadi and Allen (2011) point out, genuinely scope-ambiguous sentences as in (42) are a hallmark of the theoretical literature, not so much of naturally occurring language. In context, it is usually clear to a competent speaker which scope order is the correct one. For example, in the following two sentences, any plausible reading would have *every member* outscope *the breast* (because no two members can share the same breast), and on the other hand *the head* outscope *all U.S. intelligence agencies* (because national intelligence is a single entity, which could not be the same as multiple different entities).

(43) a. Finally the gorgeous jewel of the order, gleaming upon the breast of every member, suggested "your Badgesty", which was adopted, and the order became popularly known as the Kings of Catarrh. [GMB 72/0696][2]

b. He is the former directory of national intelligence, the head of all U.S. intelligence agencies. [GMB 59/0286]

For the GMB, it is thus neither necessary nor desirable to leave scope order underspecified. Instead, fully resolved meaning representations should be created. We thus investigated whether we could avoid the complexity of underspecified representations in the first place and treat scope compositionally, fully specifying it from the start. First, note that in the syntax-semantics interface presented in Section 5.2.4, the scope order of NPs is determined by the functors taking them as arguments, i.e., verbs and prepositions. For example, here are two possible interpretations for the transitive verb *read*, one that has the subject outscope the object (as in (42-a)), and one that has the object outscope the subject (as in (42-b)). The difference lies simply in the order in which the

[2]These codes indicate the GMB document ID the sentence is from, cf. Section 5.4.1.

generalized quantifiers that are the interpretations of the arguments are applied.

(44) a. Transitive verb, subject wide scope

$$\lambda o.(\lambda s.(\lambda m.(s@\lambda x.(o@\lambda y.(\ \boxed{\begin{array}{c}1\\ \hline 1 \leftarrow e\\ \hline \text{see.v.01}(e)\\ \text{Experiencer}(e,\, x)\\ \text{Stimulus}(e,\, y)\end{array}}\ + (m@e))))))$$

b. Transitive verb, object wide scope

$$\lambda o.(\lambda s.(\lambda m.(o@\lambda y.(s@\lambda x.(\ \boxed{\begin{array}{c}1\\ \hline 1 \leftarrow e\\ \hline \text{see.v.01}(e)\\ \text{Experiencer}(e,\, x)\\ \text{Stimulus}(e,\, y)\end{array}}\ + (m@e))))))$$

The interpretations of these functors should then be all that needs to be modified to obtain the desired readings.

To test whether this conjecture holds true in practice, we examined the data in the GMB (release 2.1.0). A pilot study showed that verb objects only rarely clearly outscope their subjects (12 of 206 instances involving a universal quantifier). We therefore focused on scope interactions mediated by prepositions. Using the syntactic annotation of the GMB, we extracted all prepositions heading a verb phrase or noun (phrase) modifier where either the object or the modified constituent contains one of the universally quantifying determiners *every*, *each* and *all*. We discarded prepositions that head verb arguments (PP/NP) because here it is the verb that mediates the scope. We discarded prepositions whose object is clausal as in *With all ballots cast, people are awaiting the results*. We also discarded prepositions that are part of fixed expressions such as *in all*, *at all*, as well as the *of* in *all of the leaders*, which we regard as part of the determiner. Finally, we cast aside the prepositions *including*, *excluding* and *except* as deserving special treatment not within the scope of this work.

This left us with 456 instances, which were annotated by one annotator for scope: should the universal quantifier have wide scope or narrow scope to get the correct reading, or doesn't it matter (e.g., because

Table 5.2: Scope interaction mediated by verb phrase modifying prepositions.

∀ in	∀ scope	example	#
obj	wide	Jobs *grew* **in** *every sector except manufacturing*, with much of the growth due to hurricane clean-up efforts in Florida. [GMB 97/0059]	57
obj	narrow	Responsibility for Africa is *currently fractured* **under** *all three*. [GMB 90/0450]	11
obj	neutral	The preferred stock, which would have a dividend rate of $ 1.76 a year, would *be convertible into Heritage common* **at** *a rate of four common shares for each preferred*. [GMB 38/0686]	3
att	wide	NATO says militants surrounded the outpost, *firing from all directions* **with** *rocket-propelled grenades, small arms and mortars*. [GMB 92/0311]	25
att	narrow	Opening batsman Jamie How *led all scorers* **with** *88 runs* as New Zealand reached 203-4 in 42.1 overs. [GMB 13/0199]	17
att	neutral	The program airs in 40 countries worldwide, and *every Idolwinner records* **through** *Sony BMG*. [GMB 75/0494]	52

the other phrase is a definite which is interpreted globally)? In cases of doubt, we preferred annotations resulting in the logically weaker reading. For example, in (45), we could assume potentially distinct termination events for each banking license or a single termination event for all of them, and we prefer the former by giving the universal quantifier wide scope.

(45) the International Banking Repeal Act of 2002 resulted in *the termination* **of** *all offshore banking licenses* [GMB 03/0688]

Table 5.2 shows the results of the annotation for verb phrase modifiers. We distinguish the following cases: the universal quantifier can occur in the modified verb phrase or in the object of the modifying preposition. In each case, it can take either wide scope (i.e., outscope

the other constituent) or narrow scope (i.e., be outscoped by it). If the modified constituent outscopes the modifier, we call this *non-inverting* scope and annotate the preposition with the `noninv` tag. Its interpretation then takes the following shape:

(46) $\lambda o.(\lambda v.(\lambda s.(\lambda m.((v@s)@\lambda e.(o@\lambda z.($ $\overset{4}{\boxed{\begin{array}{c}\\ 4 \leftarrow <\text{rel}>(e, z)\end{array}}}$ $+ (m@e)))))))$

The desired reading is thus obtained, as exemplified in Figures 5.3 and 5.4.

If the modifier outscopes the modified constituent, we call this *inverting* scope and annotate the preposition with the `inv` tag. Its interpretation then takes the following shape, applying the interpretation of the object at a higher level:

(47) $\lambda o.(\lambda v.(\lambda s.(\lambda m.(o@\lambda z.((v@s)@\lambda e.(\ $ $\overset{3}{\boxed{\begin{array}{c}\\ 3 \leftarrow <\text{rel}>(e, z)\end{array}}}$ $+ (m@e)))))))$

The desired reading is thus obtained, as exemplified in Figures 5.5 and 5.6.

Table 5.3 shows the results of the annotation for the noun (phrase) modifiers. Again, we distinguish cases by whether the modified constituent or the object of the preposition contains the universal quantifier, and by whether that takes wide scope. In the cases of non-inverting reading, i.e., where the modified constituent outscopes the modifier, we annotate the preposition with `noninv`. We obtain the desired interpretation by treating the preposition as *noun*-modifying, giving it category $(N\backslash N)/NP$, and an interpretation of the following shape:

(48) $\lambda o.(\lambda p.(\lambda x.((p@x) + (o@\lambda y.\ $ $\overset{2}{\boxed{\begin{array}{c}\\ 2 \leftarrow <\text{rel}>(x, y)\end{array}}}$ $))))$

Examples are shown in Figures 5.7 and 5.8.

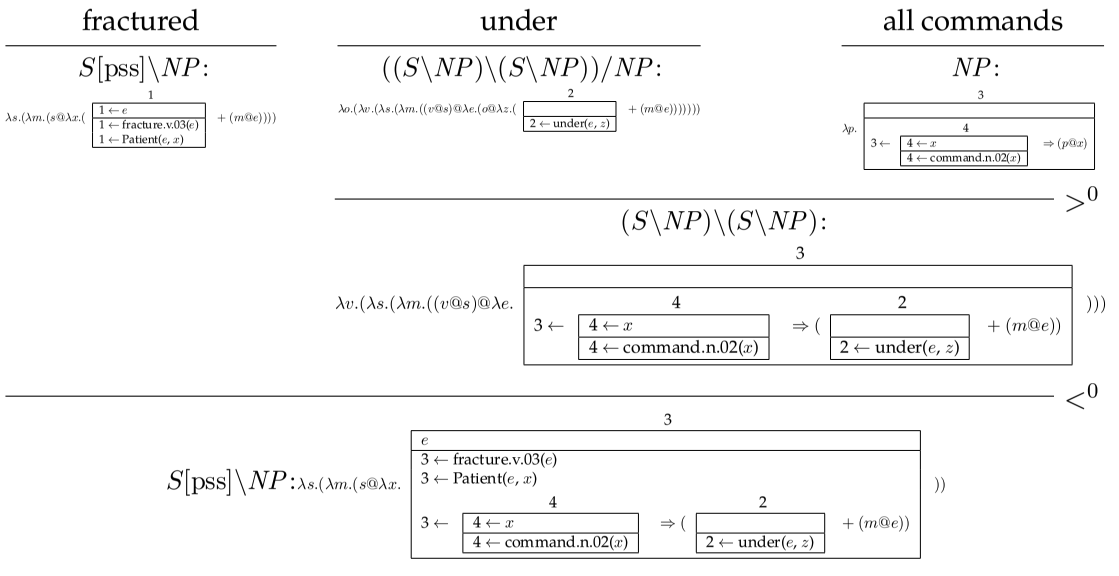


Figure 5.3: Narrow-scope universal in VP modifier.

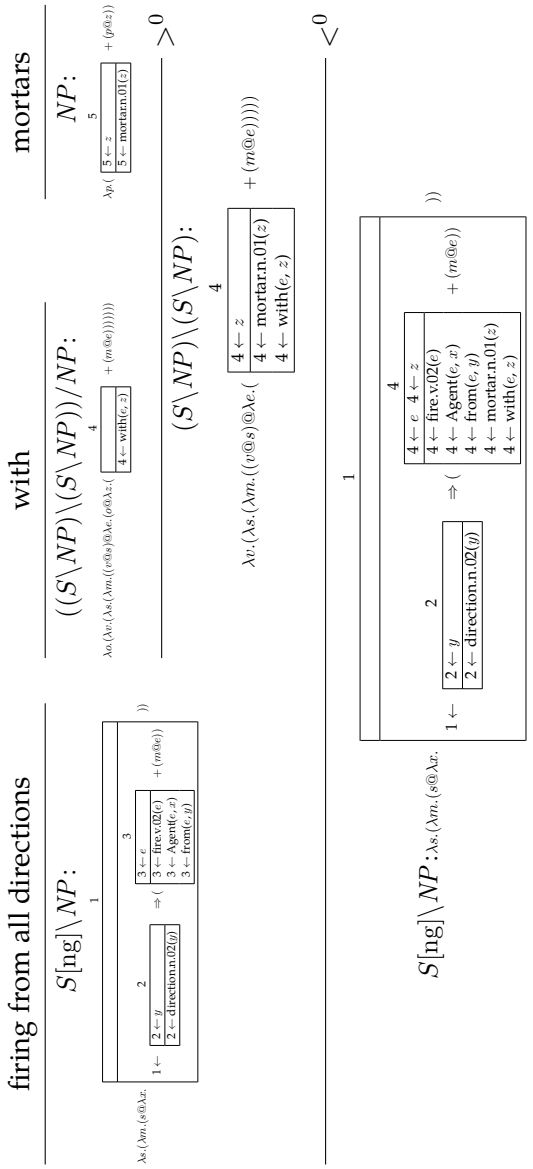


Figure 5.4: Wide-scope universal in modified VP.

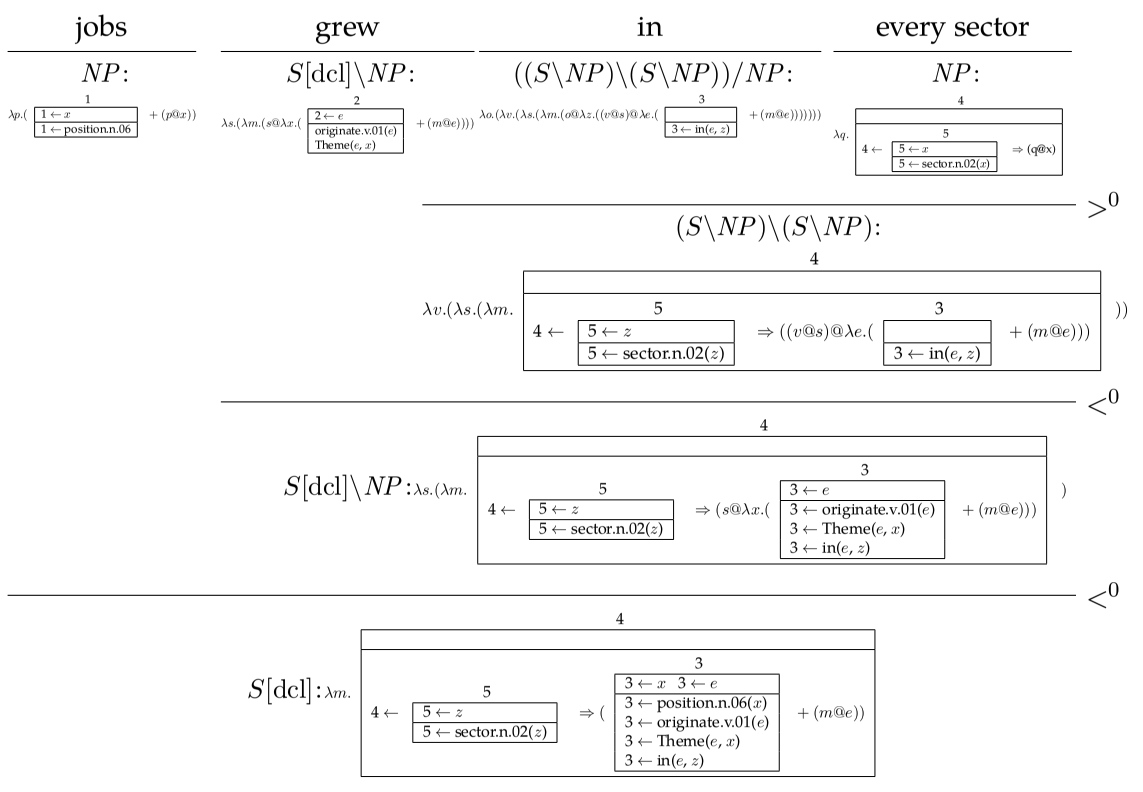


Figure 5.5: Wide-scope universal in VP modifier.

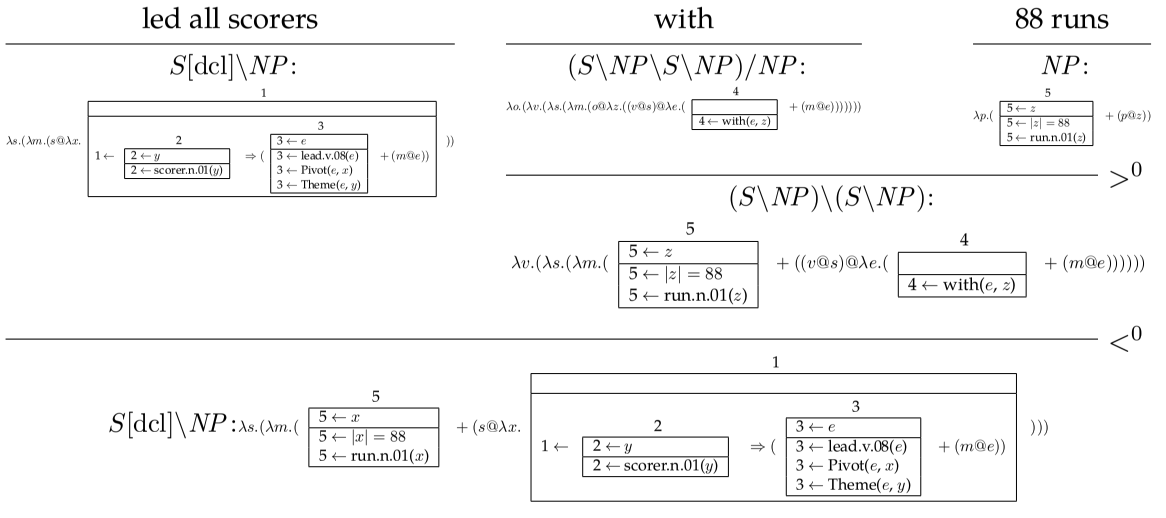


Figure 5.6: Narrow-scope universal in modified VP.

Table 5.3: Scope interaction mediated by noun (phrase) modifying prepositions.

∀ in	∀ scope	example	#
obj	wide	Finally the gorgeous jewel of the order, gleaming upon *the breast* **of** *every member*, suggested "your Badgesty," which was adopted, and the order became popularly known as the Kings of Catarrh. [GMB 72/0696]	115
obj	narrow	He is the former director of national intelligence, *the head* **of** *all U.S. intelligence agencies*. [GMB 59/0286]	44
obj	neutral	He said methods such as abortion do not fight poverty or help a country's development but actually constitute "*the destruction* **of** *the poorest of all human beings*." [GMB 13/0428]	1
att	wide	*All such attacks* **by** *drone aircraft* are believed to be carried out by U.S. forces. [GMB 76/0357]	32
att	narrow	The official Xinhua news agency says *all 28 workers* **in** *a mine in northwestern Shaanxi province* died when an underground cable caught fire on Saturday night. [GMB 40/0608]	16
att	neutral	It tacitly encouraged Iraq's minority Sunni Muslims to vote, saying *all segments* **of** *the Iraqi people* must go to the polls. [GMB 52/0038]	83

If the modifier contains a universal quantifier that outscopes the modified constituent, the discourse referent introduced by the latter must appear within the consequent of the implication. We achieve this by treating the preposition as *noun-phrase*-modifying, giving it category $(NP\backslash NP)/NP$ and an interpretation of the following shape:

(49) $\lambda o.(\lambda a.(\lambda p.(o@\lambda y.(a@\lambda x.($ $\begin{array}{|c|}\hline \\ \hline 2 \leftarrow \text{<rel>}(x, y) \\ \hline \end{array}^{2}$ $+ (p@x))))))$

An example of how the desired interpretation is derived is shown in Figure 5.9.

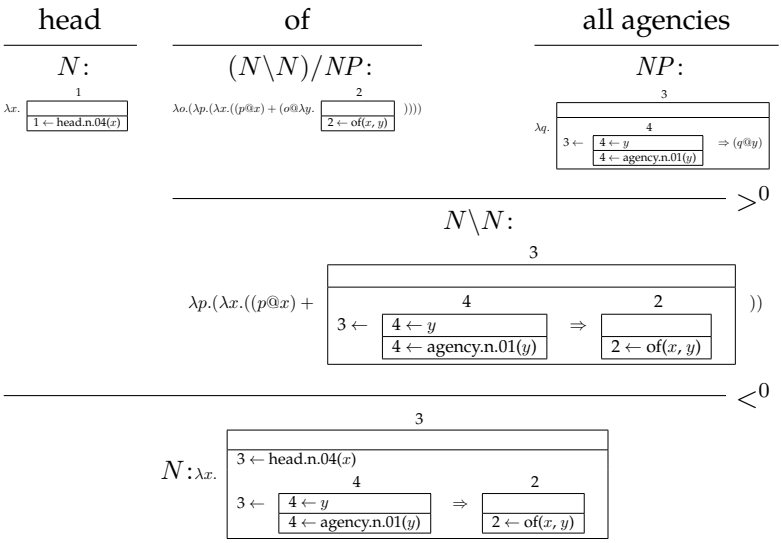
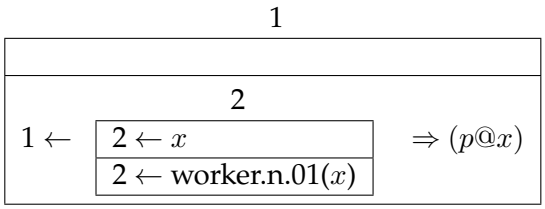


Figure 5.7: Narrow-scope universal in noun modifier.

One might expect the same strategy to work in the case of narrow-scope universals in the modified noun phrase, but this is unfortunately not the case. Recall the general form of universally quantified NPs, e.g., *all workers*:

(50)


The consequent of the implication can be determined by applying the NP to a suitable predicate p, however, the antecedent cannot be modified. This is what we would need to do in order to quantify over workers in a specific mine as in *all workers in a mine in Shaanxi*. We could get the required "in" relation condition into the antecedent by treating

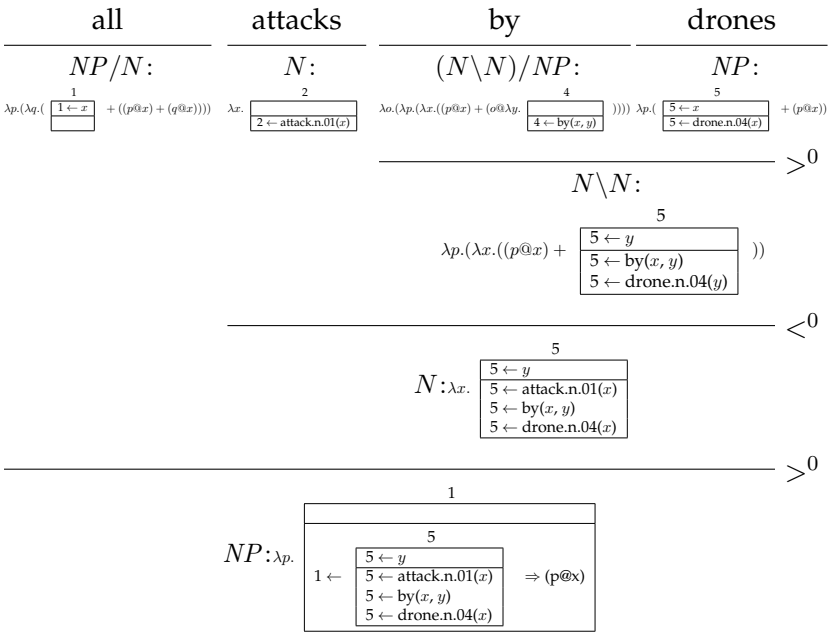


Figure 5.8: Wide-scope universal in NP with noun modifier.

the modifier as a noun modifier, but then the discourse referent for the mine would be introduced there too, failing to capture the reading that it is a *specific* mine that all workers were in. Thus, in this case, controlling scope via the prepositions that mediate it is not enough, and our syntax-semantics interface has to be adapted further to cover these cases.

Another limitation occurs when an NP A outscopes an NP B and is outscoped by NP C, where B and C both occur in the same constituent which A is not part of. We found only a single example of this in our data:

(51) *Members of an Islamic alliance and other parties took to the streets Saturday* **in** *all major cities and towns*, where speakers denounced General Musharraf for breaking his promise. [GMB 50/0210]

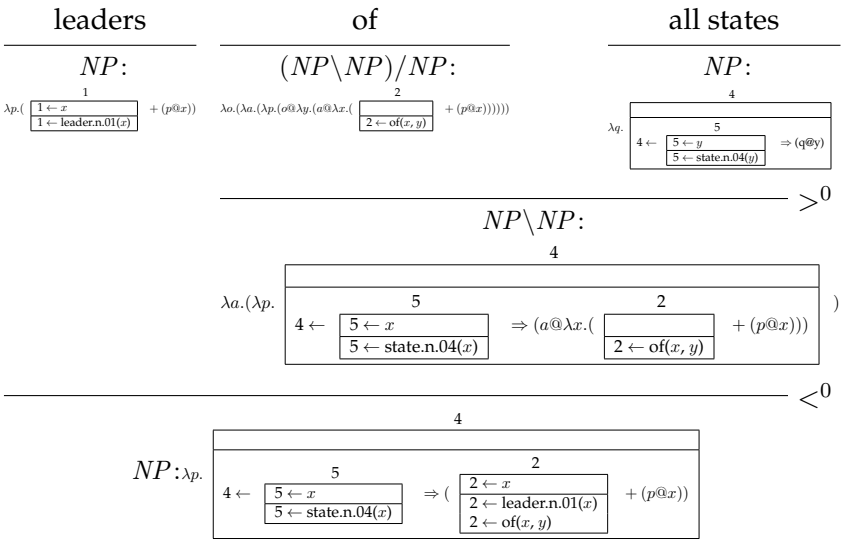


Figure 5.9: Wide-scope universal in NP modifier.

Here, *all major cities and towns* should outscope *Members* because different members presumably took to the street in each town. However, it should be outscoped by *an Islamic alliance* because that is presumably the same alliance across all towns. Our syntax-semantics interface so far unfortunately cannot represent this reading.

Of all the examined instances of quantifier scope mediated by prepositions, these problematic cases form only a relatively small part; 96% can be handled with our current framework. Future work to enhance it may draw inspiration from (Steedman, 2012). There, existentials are analyzed as skolem terms whose scope is controlled by where in the derivation an operation called skolem term specification occurs. A variant of this analysis could be developed for PDRT, and skolem term specification could be controlled by associating it with specific argument slots, similar to our lexicalized annotation of semantic roles.

Table 5.4: Subcorpora of the GMB (development version) and their sizes, as of July 2016.

Subcorpus	Genre	Documents	Sentences	Tokens
Voice of America	newswire	9,208	57,191	1,228,556
CIA World Factbook	almanac	514	4,430	111,925
Aesop's fables	fable	223	944	23,016
Jokes	humor	122	445	7,552
MASC	misc.	35	291	6,964
Total		10,103	57,191	1,228,556

5.4 Building the Groningen Meaning Bank

5.4.1 Data

To ensure wide and uncomplicated availability of the GMB, it was decided to place the annotations in the public domain, and that only documents should be included whose copyright status and license terms permit their free redistribution. A variety of genres should be included and the focus was to be on English prose, as other languages or forms such as transcribed speech or dialogue would have exceeded the scope of the annotation scheme, the available processing tools and the project. With these criteria, a total of 73,352 plain-text documents were collected from the Web sites `http://www.voanews.com`, `http://www.aesopfables.com` and `http://www.basicjokes.com` as well as from the CIA World Factbook (Central Intelligence Agency, 2006) and the Manually Annotated Subcorpus of the Open American National Corpus (Ide et al., 2010). Over the course of the project, new documents from this collection were periodically included in the GMB after reviewing them for whether they meet the criteria—filtering out, for example, documents with extensive dialogue or artifacts from Web crawling. The result is a corpus of over one million tokens in five subcorpora with different prose genres, as shown in Table 5.4.

The GMB is divided into 100 parts (00–99). Each part is represen-

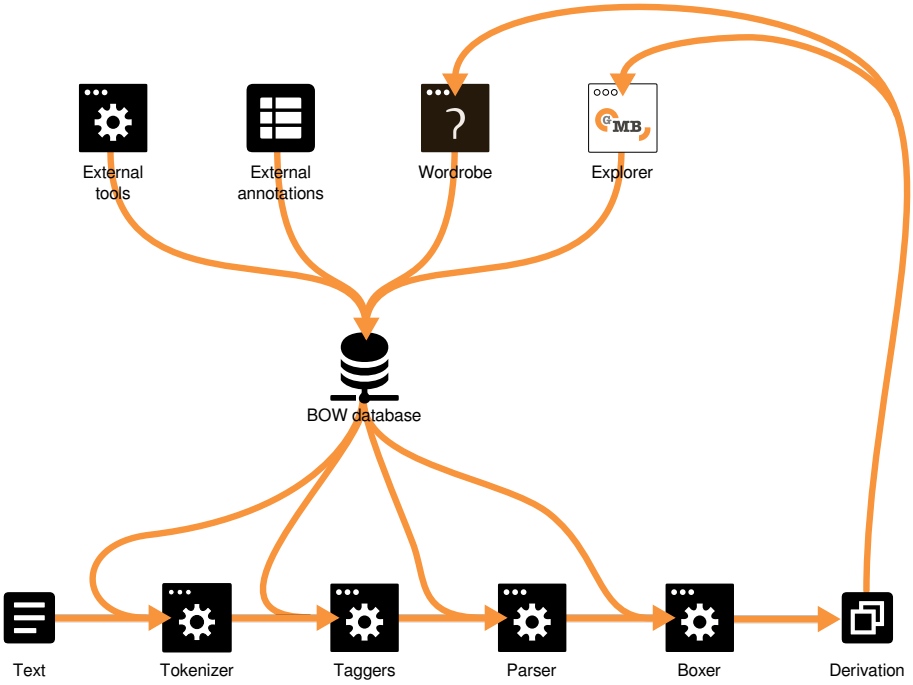


Figure 5.10: Graphical representation of the workflow for annotating the GMB. At the bottom, each tool output is corrected using any available BOWS before being sent into the next tool. The resulting derivations are fed back into Explorer and Wordrobe to solicit more annotations from experts and the crowd. Icons adapted from Thomas Helbig/The Noun Project.

tative of the whole corpus with respect to the relative sizes of the subcorpora as measured in documents. Each document is identified by a six-digit identifier starting with the part identifier. For example, document 16/0690 is the 690th document within part 16.

5.4.2 Human-aided Machine Annotation

The process of annotating the GMB takes place in a bootstrapping fashion: the raw text is first processed by a natural-language processing

toolchain to produce a complete, but not necessarily fully correct first annotation. This annotation is then gradually improved using human annotation decisions. At the core of this workflow is the toolchain depicted at the bottom of Figure 5.10. The toolchain currently consists of the following tools:

- **The statistical tokenizer and sentence boundary detector Elephant.** The Elephant text segmentation software (Evang et al., 2013) was initially trained on a small portion of gold standard data. Manual corrections to its output (see below) make the available amount of manually segmented text grow, and it is periodically retrained on this in order to learn new abbreviations or other tricky segmentation cases.

- **Various sequence taggers.** There are five taggers which label individual tokens. For POS tags, we use the part-of-speech tagger included with the C&C parser (Curran et al., 2007), trained on CCGbank (Hockenmaier and Steedman, 2007). Morphological analysis is done with morpha (Minnen et al., 2001) providing the lemma for each token. Named-entity tagging is done with the named-entity tagger included with the C&C tools, trained on the MUC data (Curran and Clark, 2003b). Animacy classification of nouns is done using the system of Bjerva (2014). For the rest of the token-level annotation decisions, we rely entirely on heuristic rules in Boxer (see below) and human annotation decisions.

- **The C&C syntactic parser.** The C&C parser is trained on CCGbank and produces CCG derivations without interpretations.

- **The semantic construction system Boxer.** Based on the output of the taggers and of the parser, Boxer (also included with the C&C parser) assigns each token a λ-DRS and outputs the resulting CCG derivations with interpretations. Where annotation is still missing for some layer, Boxer makes heuristic choices. For example, for assigning a semantic role list to a verb, Boxer uses a most-common-rolelist baseline per category.

Every component of the toolchain can, of course, make mistakes, which should then be corrected by human annotators. This poses the question how human annotation decisions should be integrated with the output of the toolchain. It is not sufficient to correct a mistake in the output of some tool (say, the part-of-speech tagger) once. The next time a mistake at an earlier stage of the toolchain (say, tokenization) is corrected, the part-of-speech tagger must be re-run because it depends on the output of the tokenizer, and the correction to the output of the part-of-speech tagger would be lost.

Our solution is to not conceptualize human annotation decisions as *changes* or *corrections*, but as *facts* or *constraints* that are permanently stored independently of the tool outputs and can be automatically applied and re-applied as needed to produce a full corrected annotation at each layer. We call these constraints "Bits of Wisdom" (BOWs). The GMB uses two types of BOW:

- A **segmentation BOW** is a pair $\langle l, tag \rangle$ where l is a character offset into the raw text, and tag is one of $\{I, O, T, S\}$, indicating the tag of the character immediately following the offset.

- A **tag BOW** is a 4-tuple $\langle l, r, layer, tag \rangle$ where l and r are character offsets in the raw text, indicating that the token that starts at l and ends at r should be tagged on layer $layer$ (e.g., $layer = pos$) with the tag tag (e.g., $tag = NNS$). In the rare event that there exists no token between the given offsets because tokenization has changed, the BOW is ignored.

Because of their use of character offsets, BOWs are instances of *stand-off annotations*. Each BOW is permanently stored in a relational database along with meta-information, such as its source, its creation time and the ID of the document it applies to. Bits of wisdom are the common currency that enables wisdom from very different sources to be accumulated in the GMB in order to obtain the best possible annotation quality. The GMB uses four sources of BOWs:

1. **The wisdom of experts.** Linguistically trained annotators can use the wiki-like Web interface of the GMB to make annotations. The

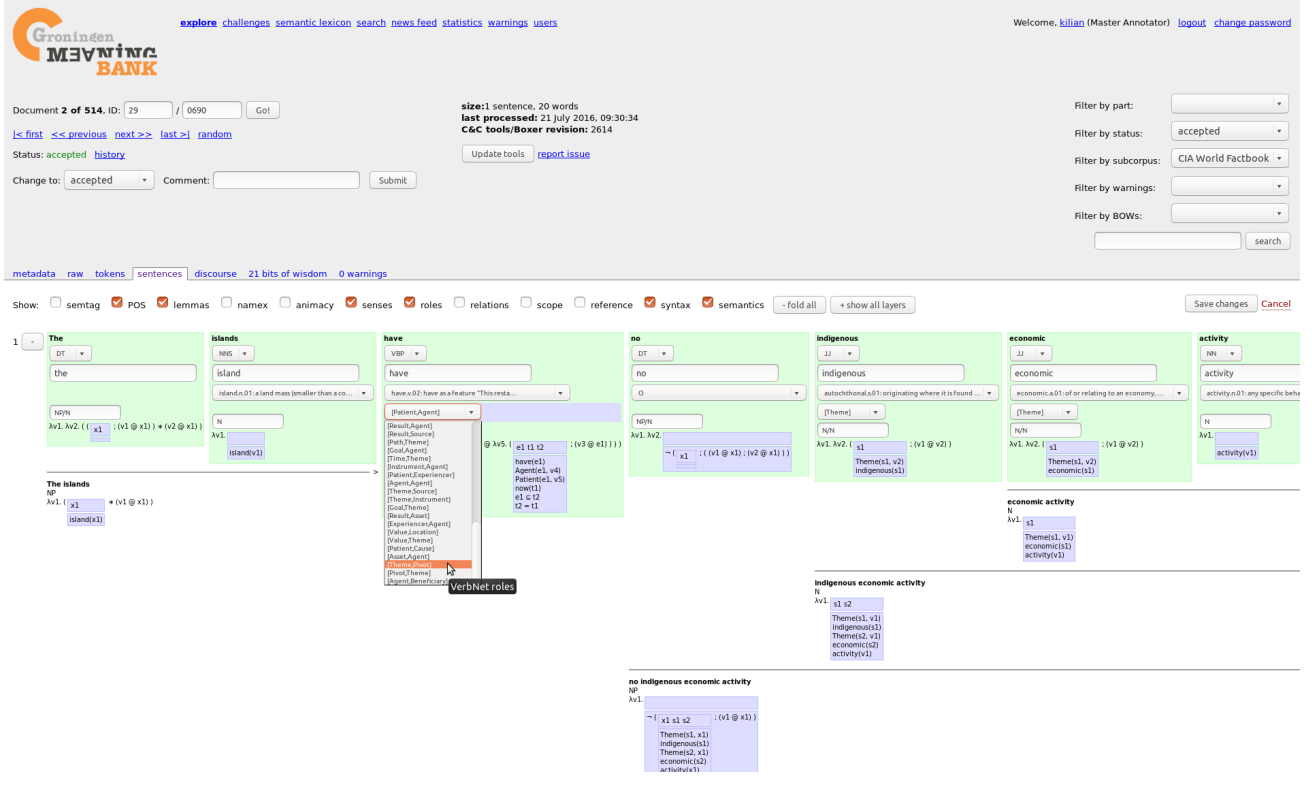


Figure 5.11: GMB Explorer, the Web interface of the GMB. At the top, there are navigation controls and filters. Different views are provided on the selected document. The "sentences" view is currently selected, showing the derivation of each sentence including token-level tags. Several annotation layers are visible and editable.

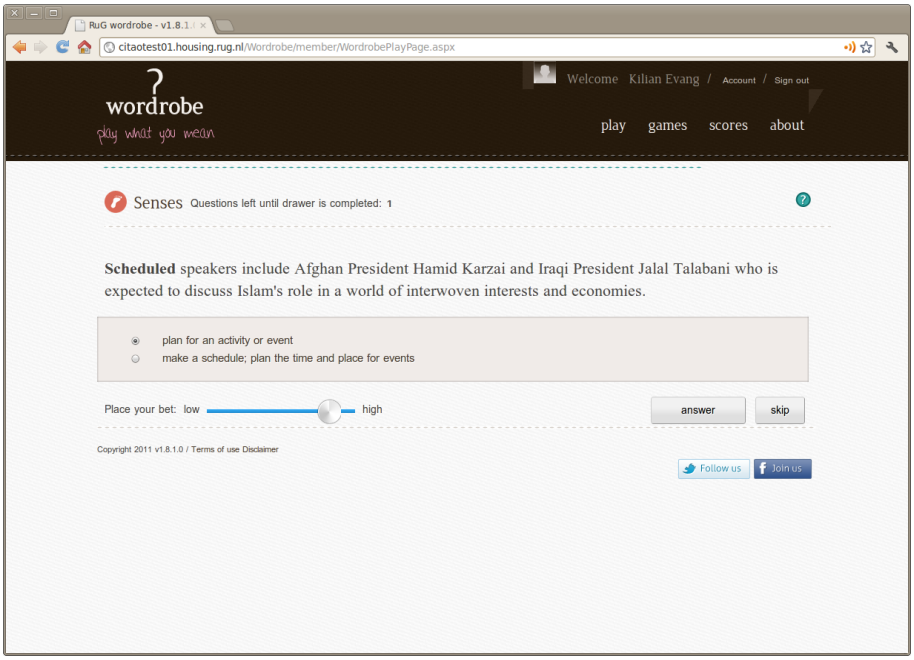


Figure 5.12: Wordrobe during a round of the *Senses* game, soliciting word sense tagging decisions from the player.

Web interface is called GMB Explorer (Basile et al., 2012a), accessible at `http://gmb.let.rug.nl/explorer/` and shown in Figure 5.11. To its users, the annotation process is presented as *editing* an annotated document, close to the traditional annotation process. However, behind the scenes, their edits are converted into BOWs and fed back into the toolchain when they click the "Save" button. Experts in the GMB team can also use scripts to add batch BOWs, addressing systematic mistakes and inconsistencies. Annotations by external contributors are monitored for quality and reverted if necessary.

2. **The wisdom of the crowd.** Many annotation decisions are crowdsourced via *games with a purpose* in which non-linguists collectively create BOWs. To this end, we developed a collection of games called

Wordrobe, shown in Figure 5.12. In a playful manner, it solicits annotation decisions about part-of-speech, word senses, named-entity types, coreference, information status, compound relations, animacy and semantic roles. BOW*s* are automatically generated from questions with high agreement among player answers (Venhuizen et al., 2013a).

3. **The wisdom of others.** One subcorpus of the GMB, MASC, has already been released with linguistic annotations by others. The released annotations were converted to BOWs and added to the GMB.

4. **The wisdom of machines.** External NLP tools can be used even without integrating them into the toolchain, by running them on the documents and converting their output into BOWs.

All BOWs are stored in the database and applied to the output of the respective tool every time the toolchain runs, as shown in Figure 5.10. A re-run of the toolchain for a document is triggered automatically when a new BOW for it is added, ensuring that the development version accessible via GMB Explorer always reflects the latest changes. Since BOWs come from different sources with varying reliability, they may conflict. The BOW application scripts therefore take the role of *judge components* that adjudicate between a set of conflicting BOWs and decide which one to apply, if any. The adjudication strategy used for the GMB is as simple as discarding crowd and batch BOWs if they conflict with another existing BOW, and applying the most recent remaining BOW.

5.5 Results and Comparison

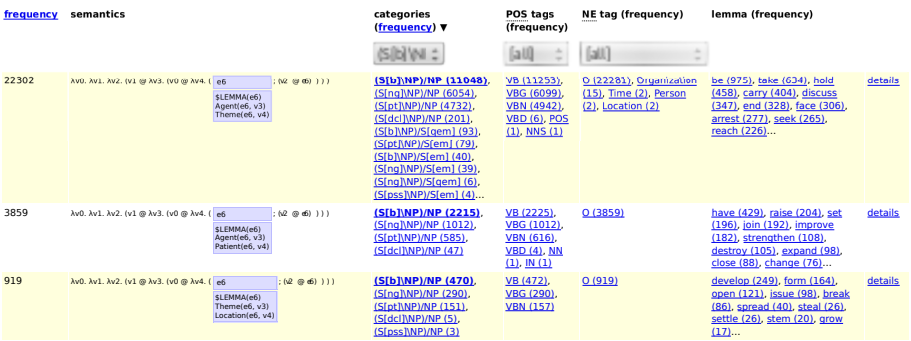
The current development version of the GMB comprises over 10,000 documents with over a million tokens. It contains the automatic annotation, improved by a total of over 170,000 BOWs (not counting those overridden by conflicting BOWs). This corresponds to roughly 1 BOW in 8 tokens. As Figure 5.13 shows, the majority of BOWs (120,340) were created by in-house expert annotators using scripts to fix systematic errors and inconsistencies. Another 42,142 BOWs were created fully manually. 4,562 BOWs so far have been aggregated from player answers in our

	expert, manual	expert, batch	crowd	others	total
tokenization	4,469	20,962	0	0	25,431
part of speech	10,634	6,725	0	6,344	23,703
lemma	804	18,234	0	0	19,038
category	405	0	0	0	405
named entity type	14,769	74,419	647	0	89,835
animacy	1,427	0	0	0	1,427
word sense	483	0	2,012	0	2,495
thematic roles	23	0	0	0	23
other relations	476	0	675	0	1,151
scope	871	0	0	0	871
co-reference	7,781	0	1,228	0	9,009
total	42,142	120,340	4,562	6,344	173,388

Figure 5.13: Active BOWs in the GMB by type and by source.

crowdsourcing application, Wordrobe. Finally, the part-of-speech annotation from MASC makes for another 6,344 BOWs. Work in progress is dedicated to creating a gold-standard portion of the GMB that has been fully checked by human annotators. There is also work on using the added annotations to retrain the toolchain and improve automatic annotation results in a bootstrapping fashion (Weck, 2015).

To date, the GMB has been used in a number of research works on semantic parsing (Le and Zuidema, 2012; Beschke et al., 2014), studying quantifier scope (Evang and Bos, 2013) and natural language generation (Basile and Bos, 2013). Another way it helps research on semantics is the *semantic lexicon* interface of GMB Explorer, which displays the list of lexical interpretations as well as counts and lists of their occurrences and co-occurrences with specific lexical categories, part-of-speech tags and named-entity tags. This interface can be used to find examples of specific linguistic phenomena, to gauge their frequencies and thus to



frequency	semantics	categories (frequency) ▼	POS tags (frequency)	NE tag (frequency)	lemma (frequency)	
22302	λv0. λv1. λv2. (v1 @ λv3. (v0 @ λv4. (e6 $LEMMA(e6) Agent(e6, v3) Theme(e6, v4) : (v2 @ e6))))	**(S[b]\NP)/NP (11048)**, (S[ng]\NP)/NP (6054), (S[pt]\NP)/NP (4732), (S[dcl]\NP)/NP (201), (S[b]\NP)/S[qem] (93), (S[pt]\NP)/S[em] (79), (S[b]\NP)/S[em] (40), (S[ng]\NP)/S[em] (39), (S[ng]\NP)/S[qem] (6), (S[pss]\NP)/S[em] (4)...	VB (11253), VBG (6099), VBN (4942), VBD (6), POS (1), NNS (1)	O (22281), Organization (15), Time (2), Person (2), Location (2)	be (975), take (634), hold (458), carry (404), discuss (347), end (328), face (306), arrest (277), seek (265), reach (226)...	details
3859	λv0. λv1. λv2. (v1 @ λv3. (v0 @ λv4. (e6 $LEMMA(e6) Agent(e6, v3) Patient(e6, v4) : (v2 @ e6))))	**(S[b]\NP)/NP (2215)**, (S[ng]\NP)/NP (1012), (S[pt]\NP)/NP (585), (S[dcl]\NP)/NP (47)	VB (2225), VBG (1012), VBN (616), VBD (4), NN (1), IN (1)	O (3859)	have (429), raise (204), set (196), join (192), improve (182), strengthen (108), destroy (105), expand (98), close (88), change (76)...	details
919	λv0. λv1. λv2. (v1 @ λv3. (v0 @ λv4. (e6 $LEMMA(e6) Theme(e6, v3) Location(e6, v4) : (v2 @ e6))))	**(S[b]\NP)/NP (470)**, (S[ng]\NP)/NP (290), (S[pt]\NP)/NP (151), (S[dcl]\NP)/NP (5), (S[pss]\NP)/NP (3)	VB (472), VBG (290), VBN (157)	O (919)	develop (249), form (164), open (121), issue (98), break (86), spread (40), steal (26), settle (26), stem (20), grow (17)...	details

Figure 5.14: Excerpt from the semantic lexicon showing the most frequent entries for category $(S[dcl]\backslash NP)/NP$.

prioritize efforts for further annotation tool development. For example, Figure 5.14 shows the three most frequently used semantic lexical entries for category $(S[\mathrm{dcl}]\backslash NP)/NP$ as displayed in this interface.

Several other broad-coverage resources of English texts annotated with formal meaning representations ("sembanks") have been released in recent years, to wit Le Petit Prince in UNL (Martins, 2012), the Treebank Semantics Corpus (Butler, 2015), the UCCA corpus (Abend and Rappoport, 2013) and AMRbank (Banarescu et al., 2013).[3] Each comes with its own meaning representation language and all differ in which aspects of meaning are captured. We highlight some of these differences in the following.

Anchoring In the Treebank Semantics corpus and in the GMB, the possible meaning representations for a sentence are constrained by the surface form of the sentence. Each meaning representation is based on and explained by a syntactic analysis of the sentence, rather than be-

[3] LinGO Redwoods (Oepen et al., 2002), DeepBank (Flickinger et al., 2012a), ParDeepBank (Flickinger et al., 2012b) and ParGramBank (Sulger et al., 2013) may be considered sembanks, too, because they include Minimal Recursion Semantic representations which are somewhat "deeper" than syntactic phrase structure trees or dependency graphs. However, since they only treat "compositional" aspects of meaning in the sense of Bender et al. (2015), and not, for example, word senses, we classify them as a syntactic resources for the present purpose.

ing created by annotators "from scratch". Advantages are that it allows to automate more of the annotation process, may lead to more consistent representations and provides a richer starting point for building semantic parsers. The disadvantage is that meaning representations are limited by the capabilities of the syntax-semantics interface used. For example, syntactically unusual constructions that may have straightforward semantic representations cannot be so annotated if, say, the lexicon and the grammar are not prepared to deal with them. UNL and AMR opt to be agnostic about how strings and meaning representations are related, and create the latter independently from the former. UCCA is an interesting hybrid. On one hand, its meaning representation graphs are created by annotators without being constrained by a grammar, and they do not correspond to constituents but to semantic objects such as *scenes* or *participants*. On the other hand, the leaves of each graph are the nodes of the sentence, and they are either used to represent concepts/relations or they are explicitly marked as purely functional elements.

Concepts All the resources except the UCCA corpus represent the meanings of content words using concepts of some ontology—an important aspect considering the polysemy and synonymy found in natural language. UNL, Treebank Semantics and GMB use WordNet senses. AMR uses a mix of English words, Wikipedia URLs and framesets from PropBank (Palmer et al., 2005). UCCA does not represent concepts on its foundational layer, although its formalism allows for adding them on higher layers in the future.

Roles Similar to concepts, semantic roles are important for capturing semantic commonalities between different forms, e.g., the semantic equivalence of the sentences "Mary was kidnapped by pirates" and "Pirates kidnapped Mary". Again, UCCA does not represent roles on its foundational layer, although its formalism allows for adding them in the future. All other resources have their own custom inventory of roles, with the GMB heavily drawing on VerbNet/LIRICS and AMR, on PropBank.

Semantic Normalization Useful semantic equivalences can hold between constituents of different types, e.g., between the noun phrase *Sam's destruction of the city* and the clause *Sam destroyed the city*. This becomes especially clear when one attempts to find a single meaning representation for a set of sentences in different languages which are translations of each other, as the way things are expressed syntactically often changes in translation. AMR is unique in that it actively tries to resolve such categorially different syntactic forms to the same meaning representations, representing meaning of content words with *frames* wherever applicable. By contrast, through their use of WordNet synsets, UNL, Treebank Semantics and GMB are committed to a representation that represents noun, verb, adjective and adverb meanings through distinct sets of concepts, retaining a certain amount of dependency of the meaning representation on "syntactic idiosyncrasies" (Banarescu et al., 2013). In the case of the GMB, two broad strategies could be employed to overcome this problem in the future: i) adopting a common set of concepts for verbs, adjectives, event nouns, relational nouns, etc., similar to AMR, ii) recognizing equivalences through reasoning, with world knowledge axioms à la Ovchinnikova (2012). UCCA, again, is a special case: as discussed, its foundational layer does not map content words to concepts at all. However, it strongly supports cross-categorial semantic normalization because its meaning representations are organized around *scenes* (similar to frames) which can be anchored to any part of speech.

Depth The meaning representations of Treebank Semantics and the GMB come with explicit model-theoretic interpretations and can be converted to first-order logic formulas, enabling approaches to natural language understanding that are based on automated reasoning. Although conversion procedures could be invented for UNL, UCCA and AMR, these were not designed with such a goal in mind. In particular, they do not explicitly represent "deep" aspects of meaning like universal quantification or not-at-issue content (handled in the GMB as projected content).

Gold Standard Le petit prince in UNL, the UCCA corpus and AMR-bank all take the usual route of releasing only annotations that have been created or checked by humans—a gold standard. In the Treebank Semantics corpus and all GMB releases to date, annotations are only silver-standard quality, improved by more human annotation decisions from release to release, but no portion yet fully corrected on all annotation layers.

5.6 Conclusions

We have described the Groningen Meaning Bank (GMB), an effort to build a large-scale corpus of English texts annotated with deep meaning representations. We argued that "human-aided machine annotation" is needed to build a corpus of such size and depth of annotation, and that the *bits of wisdom* approach is an effective framework for combining the knowledge of machines, human experts, human non-experts and external resources. We also showed that even phenomena traditionally dealt with at a non-lexical level, in particular quantifier scope, can largely be treated with a fully lexicalized mode of annotation.

It is probably fair to say that in comparison to other semantic annotation efforts, the GMB employs the "deepest" meaning representation formalism, with a detailed representation of, e.g., universal quantification and not-at-issue content. It is an especially rich resource also because the text-meaning pairs are anchored in explicitly given CCG annotations, and many additional layers of annotation are explicitly included. On the other hand, the GMB is further from the goal of producing fully gold-standard data than most comparable projects, which limits its usefulness as training and testing data for machine learning.

We conclude that although the "human-aided machine annotation" approach is useful for rapidly developing a complex annotation formalism and methodology, and testing it on large amounts of data, additional focused human annotation efforts will be required for obtaining large amounts of gold standard data. In the meantime, there is a trade-off between logical depth, dataset size and gold standard quality.

The GMB focuses exclusively on English. If other languages are to be

annotated with deep meaning representations, does one have to make a new corpus from scratch? Or can parts of the existing annotations be reused for other languages? The latter may be possible if we can automatically project CCG derivations from one language to another. We will investigate this in the next chapter.

Chapter 6

Derivation Projection Theory

6.1 Introduction

We now turn to the goal of creating broad-coverage semantic parsers for arbitrary target natural languages with cross-lingual supervision. Specifically, the only training data we wish to assume access to is parallel data, pairing target-language sentences with translations in the source language, where source-language sentences have already been semantically parsed and word-aligned with the target-language sentences.

For now, we will assume that the source-language semantic parsing and word alignment has gold-standard quality, for example as the result of an annotation process as described in Chapter 5. In Chapter 7, we will explore the possibility of doing this automatically, requiring the method to be robust to noise.

6.1.1 Cross-lingual Semantic Annotation

To train a target-language semantic parser, we need the target-language sentences to be annotated with meaning representations. If the sentence pairs in our parallel corpus have the same meaning, we can just copy the meaning representation from the source side to the target side. Given that the sentences are translations of each other, the assumption that they have the same meaning should be justified in general.

Of course, there are exceptions. Sometimes, one sentence is more specific than another. For example, the English word *player* may be translated with the German word *Fußballspieler* (literally: soccer player). Other times, the translation is both more and less specific. For example, for the phrase *a pint of beer*, the German translation *ein Glas Bier* (literally: a glass of beer) is more specific with respect to the material, but less specific with respect to the size of the container. Bos (2014) calls these types of divergent translations *informative translations* and *loose translations*, respectively. We ignore them in this chapter and treat them as an additional source of noise that our eventual system will have to deal with robustly.

An additional source of divergent meanings may be idioms, if represented literally. For example, *kick the bucket* and the German *den Löffel abgeben* (literally: to surrender the spoon) are perfectly adequate translations of each other, although their "literal" meanings are completely different. Ideally, the meaning representation should abstract from the literal meaning and only contain the metaphorical one, e.g., using the WordNet sense die.v.01. However, in practice, many annotation methodologies and semantic parsers are not yet able to do this, so this is another issue we have to be aware of and deal with when using source-language meaning representations to train target-language semantic parsers.

6.1.2 Cross-lingual Grammar Induction

In order to get a target-language semantic parser, we also need some form of grammar for the target language. As seen in Chapter 3, some narrow-domain semantic parsers are able to induce grammars with few prior assumptions about the target-language syntax. This would be ideal for our goal of being able to process arbitrary target languages without prior knowledge about them. Unfortunately, to the best of our knowledge, such methods have not been shown to scale to broad-coverage tasks. All broad-coverage semantic parser learners we know of make use of an existing, explicitly supervised syntactic parser for the target language (cf. Section 3.7).

Our proposed solution to this problem is to transform the existing source-language parses into parses for the target-language translations

automatically. That is, we would like to automatically create a parallel treebank, from which a target-language grammar can be read off.

Of course, every language has a different grammar. Can the source-language parses possibly contain any useful information about what the target-language parses should be like? Indeed, there is precedent to suggest that this should be the case, *viz.* the success of (manually built) parallel grammars like ParGram (Butt et al., 2002) and (manually built) parallel treebanks like ParGramBank (Sulger et al., 2013). An explicit aim of these projects is to assign sentences which are translations of each other analyses that are as similar as possible and as different as necessary. To do this, they use Lexical Functional Grammar (LFG; Bresnan, 2001; Dalrymple, 2001), which is designed around a version of the Universal Grammar hypothesis (Chomsky, 1988, 1995), namely that "all languages are structured by similar underlying principles" (Butt et al., 2002). An LFG analysis of a sentence consists of an f-structure, which is meant to encode language-independent aspects of a sentence's structure such as predicate-argument structure and statement type, and a c-structure, which encodes language-specific aspects such as word order, constituency or morphological vs. syntactic structures. Grammars consist of a mapping from sentences to c-structures and a mapping from c-structures to f-structures. Thus, in ParGramBank, to a large extent, f-structures are identical between translations.

The success of such "maximally parallel" grammars and treebanks suggests it may be possible to transfer the language-independent aspects of an analysis from a sentence to its translation, and discover the language-specific aspects automatically, learning them from the data. In this chapter, we address the question if, and how, this can be done, by proposing a series of algorithms to this end, each more capable than the previous, and discussing their capabilities and limitations.

We base our method on Combinatory Categorial Grammar mainly for three reasons. First, our existing tools for parsing text into Discourse Representation Structures are based on CCG, so we could then reuse them for cross-lingual semantic parsing. Secondly, CCG's close coupling of syntax and semantics makes the integration of meaning representations into the projection process straightforward. Thirdly, we believe that CCG is particularly well suited to representing the syntax

of translations with maximum parallelism. This is because its combinatory rules are designed to be universal, leaving lexical entries as the only formal objects that need to differ between languages.[1] Also, categories closely correspond to semantic types, often abstracting over differences such as adverbs vs. prepositional phrases as modifiers. As we develop our method, we will also seek an answer to the question in how far our belief in CCG as a suitable formalism for parallel representation is correct.

The basic versions of our algorithm are presented in Section 6.2, more advanced ones in Section 6.3. The treatment of source-language words not corresponding to any target-language words is discussed in Section 6.4. We evaluate our method in Section 6.5 by checking its capabilities against a list of known types of divergences between sentences and their translations. We formulate the answers to the above research questions in Section 6.6.

6.2 Basic Derivation Projection Algorithms

The input to our proposed projection algorithm is a source-language sentence $\mathbf{e}$ with a CCG derivation and its target-language translation $\mathbf{f}$. The output is a derivation for $\mathbf{f}$, with the same interpretation as $\mathbf{e}$.

The basic idea is to assign the words in $\mathbf{f}$ the same interpretations and the same categories as the corresponding words in $\mathbf{e}$ and then build a derivation, guided by the given interpretation. In the following, we motivate each part of the algorithm and describe a series of approximations to it, with increasing capabilities. Many of our example sentences are drawn from the Parallel Meaning Bank (PMB), a multilingual successor project to the Groningen Meaning Bank (Bjerva et al., 2014). We use English as the source-language in all examples as this is the primary source language we wish to apply the process to.

[1] In practice, type-changing rules and rule restrictions may also differ. However, in statistical parsing, the latter play less of a role, as discussed in Section 2.4.6.

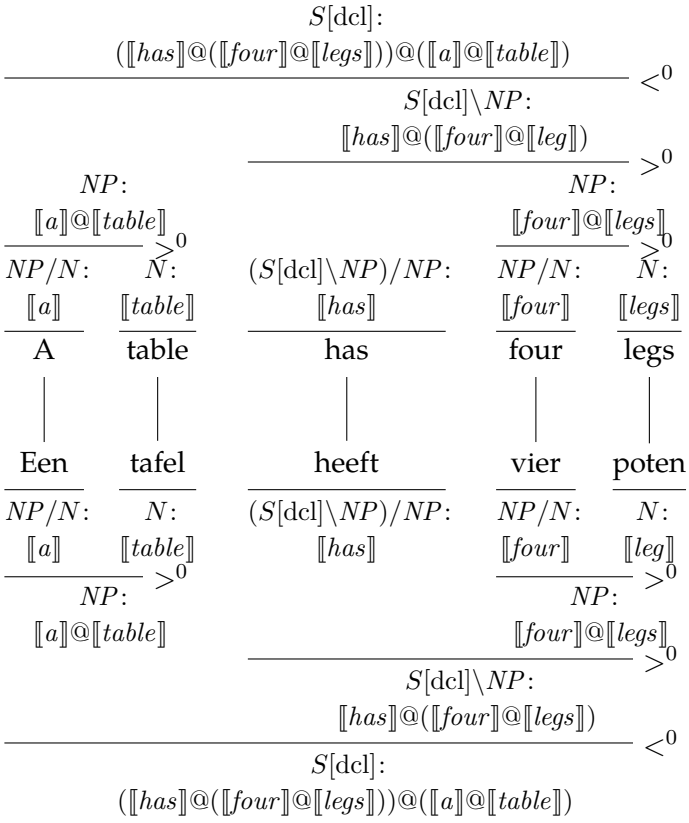


Figure 6.1: Example of an English CCG derivation, a word-aligned Dutch translation and a parallel Dutch derivation. The English derivation is given on top, vertically mirrored compared to the usual way of drawing. The Dutch derivation is given below, not mirrored. Word interpretations depend on the semantic formalism used; we write them using the $[\![\cdot]\!]$ notation. [PMB 65/0788]

6.2.1 TRAP: Transfer and Parse

Our first algorithm, called TRAP, is stunningly naïve. It proceeds as follows:

1. TRANSFER: Assign the i-th word in $\mathbf{f}$ the category and interpretation of the i-th word in $\mathbf{e}$.

2. PARSE: Apply combinatory rules to obtain a normal-form derivation for $\mathbf{f}$ that has the same interpretation as that for $\mathbf{e}$.

TRAP gives the desired results for sentence pairs that have the same number of words and moreover have a word-for-word correspondence between both sentences, as in the example shown in Figure 6.1. This is not much, of course. In these extremely simple cases, we could achieve the same effect by copying the English derivation verbatim and just substituting the target-language words for the source-language words. TRAP instead uses a two-step approach where the PARSE step builds up target-language derivations from the ground. This allows for changing the way in which categories are transferred in the more sophisticated algorithms building on TRAP.

The PARSE step is subject to two constraints: first, the derivations we build must be normal-form (Eisner, 1996; Hockenmaier and Bisk, 2010). This is standard practice in CCG parsing and prevents us from using higher-degree composition rules and type-raising rules where they are not needed, which would lead to many possible semantically equivalent derivations with spurious syntactic differences. The second constraint is, of course, that the resulting derivation must assign the same semantics to $\mathbf{f}$ as is assigned to $\mathbf{e}$.

The second constraint could be enforced by building all derivations possible with the lexical items assigned by the transfer step and CCG's combinatory rules, and then discarding those whose interpretation is not equivalent to that of $\mathbf{e}$. In practice, the search space would often be too large for this. We should therefore immediately discard any intermediate constituent where we can tell from looking at its interpretation that it could not possibly lead to the desired interpretation. An

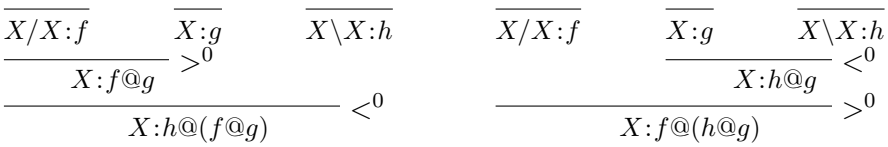


Figure 6.2: A simple example of two derivations with the same lexical constituents but different interpretations.

abstract example is shown in Figure 6.2, where the same lexical constituents lead to two derivations with different interpretations. Let us say that $h@(f@g)$ is the correct interpretation, then already the intermediate constituent in the right-hand side derivation with interpretation $h@g$ is surely incorrect. $h@g$ cannot lead to $h@(f@g)$, it is not a "valid semantic fragment" of it. Let us now make this notion precise.

Observe that in CCG's combinatory rules (cf. Section 2.4), the interpretations of all input constituents always occur as a subterm of the interpretation of the output constituent. We could thus simply say that an interpretation I^* is a valid semantic fragment of an interpretation I iff I^* is a subterm of I. However, we have to allow for the effects of α-conversion and β-conversion, which do not preserve all subterms (cf. Section 2.2).

Specifically, they preserve all subterms with the following exceptions:

1. When a term $(\lambda x.M)$ is α-converted, the result does not contain any of its subterms containing x; it contains subterms containing another variable instead.

2. When a term $((\lambda x.M)@N)$ is β-converted, the result does not contain the $((\lambda x.M)@N)$ or $(\lambda x.M)$ subterms. Furthermore, the result does not contain any of the subterms of M containing x; it contains subterms containing N instead.

3. When a term $((\lambda x.M)@N)$ is β-converted and x does not occur in M, the result does not contain N.

We need not worry about the last case because for the purposes of the PARSE step, we treat all interpretations of source-language words, such as $[\![table]\!]$, as free variables, so they cannot contain functions that discard their input. Furthermore, the combinatory rules do not generate such functions. Allowing for disappearance of certain subterms according to the first two cases, we can still be sure that certain subterms of I^* have equivalents in I.

We assume that I and I^* are in normal form, i.e., they do not contain any terms of the form $((\lambda x.M)@N)$. Let $I^* = (\lambda x_1.(\lambda x_2.\cdots(\lambda x_n.M)\cdots))$ where $n \geq 0$. The subterms $x@N$ in M where $x \in \{x_1, x_2, \cdots, x_n\}$ are the places where β-conversion can happen when I^* is applied to arguments later in the derivation. We call these subterms the *volatile subterms*. We then define as a *stable subterm* of I^* any subterm of M which does not contain a volatile subterm. Stable subterms can still contain bound variables which are later replaced by other terms, but only in argument positions, so that no additional β-reduction outside the new terms becomes possible. Therefore, every stable subterm J^* of I^* must *subsume* some subterm J of I, meaning that that there are terms $B_1, B_2, \cdots, B_n$ such that $J^*[x_1 := B_1][x_2 := B_2]\cdots[x_n := B_n] \equiv J$.

We use a function vsf ("valid semantic fragment") that checks this for all maximal stable subterms (it follows for all non-maximal stable subterms). Specifically, $vsf(I^*, I) = 1$ iff all maximal stable subterms of I^* subsume some subterm of I. If this is not the case for the interpretation of some constituent, we discard it immediately.

6.2.2 ARTRAP: Align, Reorder, Transfer and Parse

TRAP's most glaring defect is that it assumes words in $\mathbf{f}$ are in the same order as the corresponding words in $\mathbf{e}$. To fix this and to ensure we transfer corresponding categories and interpretations from $\mathbf{e}$ to $\mathbf{f}$ in the transfer step, we add a reordering step before TRANSFER in which the $\mathbf{e}$ words are brought into the order corresponding to $\mathbf{f}$. For this we must know which word corresponds to which. Let us, for now, assume that we can obtain sound knowledge about this, e.g., from human annotators or from word alignment tools developed in the field of Machine Translation. The algorithm then becomes:

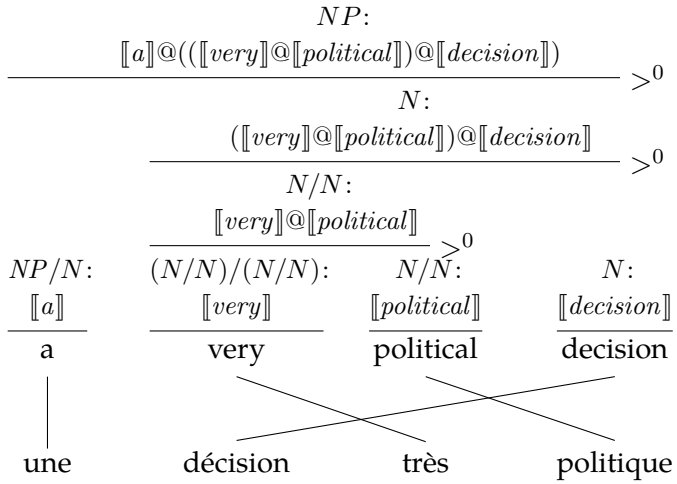
1. Align: Determine which word in **e** corresponds to which word in **f**.

2. Reorder: Stably sort the words in **e** so they appear in the same order as the corresponding words in **f**.

3. Transfer: Assign the i-th word in **f** the category and interpretation of the i-th word in the reordered sentence **e**.

4. Parse: Apply combinatory rules to obtain a normal-form derivation for **f** that has the same interpretation as that for **e**.

Consider, for example, the following derivation of a noun phrase and the word alignment with the French translation:



It shows what is a frequent phenomenon in comparing English and French: attributive adjectives normally precede the noun they modify in English, but normally succeed it in French (Dryer, 2013c). The order of attributive adjective and noun is only one example of a vast variety of word order features that differ systematically across languages and that can, but do not have to be, mostly consistent within any one language

(Dryer, 2013a). Other such features include the order of subject, object and verb (Dryer, 2013f), that of demonstrative and noun (Dryer, 2013d) or that of relative clause and noun (Dryer, 2013e). It is therefore crucial that our algorithm can handle these differences.

The Reorder step produces the following output in our example:

NP/N: $[\![a]\!]$	N: $[\![decision]\!]$	$(N/N)/(N/N)$: $[\![very]\!]$	N/N: $[\![political]\!]$
a	decision	very	political

The Transfer step then assigns categories and interpretations as follows:

une	décision	très	politique
NP/N: $[\![a]\!]$	N: $[\![decision]\!]$	$(N/N)/(N/N)$: $[\![very]\!]$	N/N: $[\![political]\!]$

The interpretations are correct, but the Parse step will fail now because the adjective *politique* will not be able to combine with the noun it is supposed to modify: the slash in its category N/N is leaning the wrong way. So Artrap does no better than Trap. Another refinement is needed.

6.2.3 Arftrap: Align, Reorder, Flip, Transfer and Parse

Our next algorithm, Arftrap, extends Artrap with another step after reordering: the slashes in the syntactic categories are flipped as needed. More precisely, whenever reordering changed the direction in which an argument is positioned relative to its functor, the slash in the category of the functor corresponding to that argument is flipped—not only on the functor itself, but also in all instances that are structure-shared (cf. Section 2.5) in the derivation for e. This is the new algorithm:

1. ALIGN: Determine which word in **e** corresponds to which word in **f**.

2. REORDER: Stably sort the words in **e** so they appear in the same order as the corresponding words in **f**.

3. FLIP: In the syntactic categories, replace all structure-shared instances of / (\) corresponding to arguments which were moved to the other side of their functor into \ (/).

4. TRANSFER: Assign the i-th word in **f** the category and interpretation of the i-th word in the modified sentence **e**.

5. PARSE: Apply combinatory rules to obtain a normal-form derivation for **f** that has the same interpretation as that for **e**.

After FLIP, the output of TRANSFER in our example now is:

une	décision	très	politique
NP/N:	N:	$(N\backslash N)/(N\backslash N)$:	$N\backslash N$:
$[\![a]\!]$	$[\![decision]\!]$	$[\![very]\!]$	$[\![political]\!]$

This finally enables PARSE to build the correct French derivation:

une	décision	très	politique
NP/N:	N:	$(N\backslash N)/(N\backslash N)$:	$N\backslash N$:
$[\![a]\!]$	$[\![decision]\!]$	$[\![very]\!]$	$[\![political]\!]$

$$\frac{[\![very]\!]\ \ [\![political]\!]}{[\![very]\!]@[\![political]\!]:\ N\backslash N}\ >^0$$

$$\frac{[\![decision]\!]\quad [\![very]\!]@[\![political]\!]:\ N\backslash N}{([\![very]\!]@[\![political]\!])@[\![decision]\!]:\ N}\ <^0$$

$$\frac{[\![a]\!]\quad ([\![very]\!]@[\![political]\!])@[\![decision]\!]:\ N}{[\![a]\!]@(([\![very]\!]@[\![political]\!])@[\![decision]\!]):\ \ }\ >^0$$

Apart from the differences in order, slashes and words, the French derivation is trivially different from the English one in that one instance of forward application has been replaced by backward application. But ARFTRAP can also handle cases that require more complex structural changes. For example, not only does it happen that an entire argument moves to the other side of its functor, it can also be only part of an argument, in which case PARSE must introduce crossed composition.

Causes of this include cases where an adverb is VP-final in English but VP-internal in German as, in Figure 6.3, or VPs that are nested in English but cross-serial in Dutch (Huybregts, 1976), as in Figure 6.4.

The opposite case, where the source language has a crossed construction but the target language sentence does not, is exemplified in (1), where English exhibits Heavy NP Shift (Ross, 1967). The derivation for the Dutch (1-b) most closely matching the structure for that of the English (1-a) would involve harmonic composition where (1-a) uses crossed composition, but since we enforce normal-form derivations, ARFTRAP will only use application here.

(1) a. I gave to my friend this beautiful gift.
b. Ik gaf dit mooie kado aan mijn vriend.

Note also how ARFTRAP can handle thematic divergences. A *thematic divergence* (Dorr, 1993) between source and target language sentence occurs when corresponding words fill different syntactic roles. For example, the subject of a verb in the source language may correspond to the object in the target language, and vice versa, as exemplified in Figure 6.5. One linguistically motivated approach would be to give *gustan* the same semantics as *like*, but with the argument order reversed: $\lambda e.\lambda s.(\llbracket like \rrbracket @ s) @ e$. This would allow the verb to combine with its arguments in the usual object-subject order and still produce the correct semantics. However, ARFTRAP is unaware of the syntactic roles of the target language, so it cannot detect when such a change is appropriate. It just makes sure the same sentence semantics is obtained and does not change the internal structure of lexical entries. In cases such as the present example, this results in the verb combining with its subject first and object second. In this particular example, no flipping is even neces-

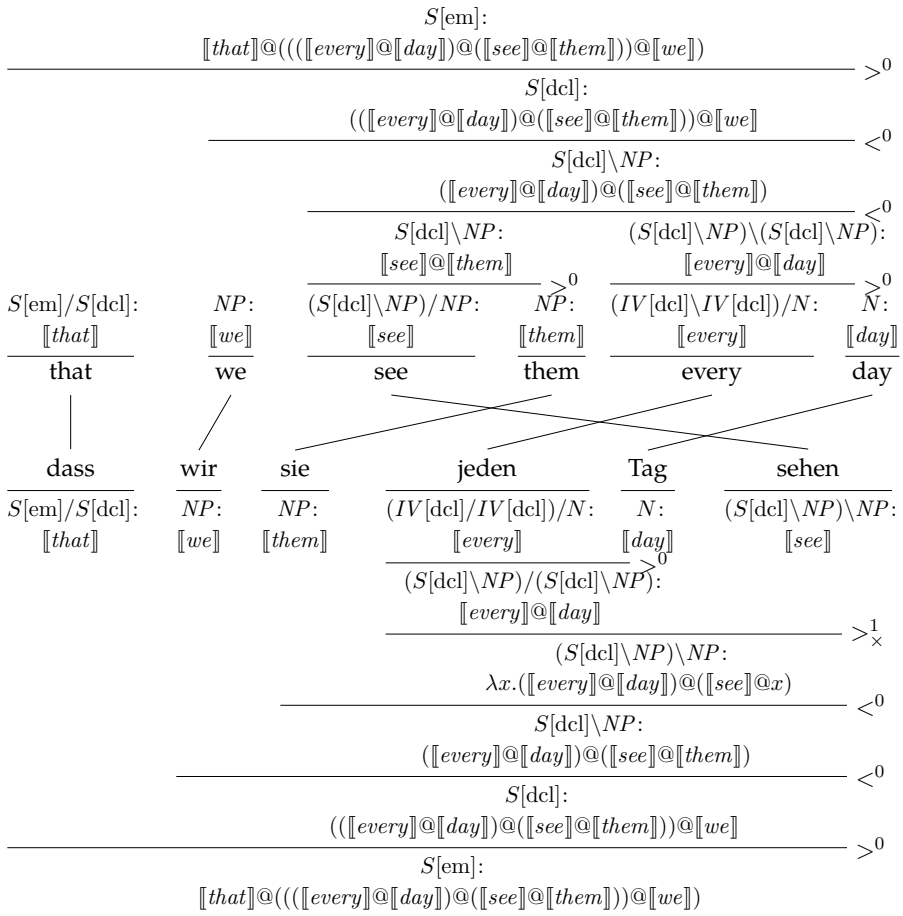


Figure 6.3: VP-internal modification in German requires the use of crossed composition.

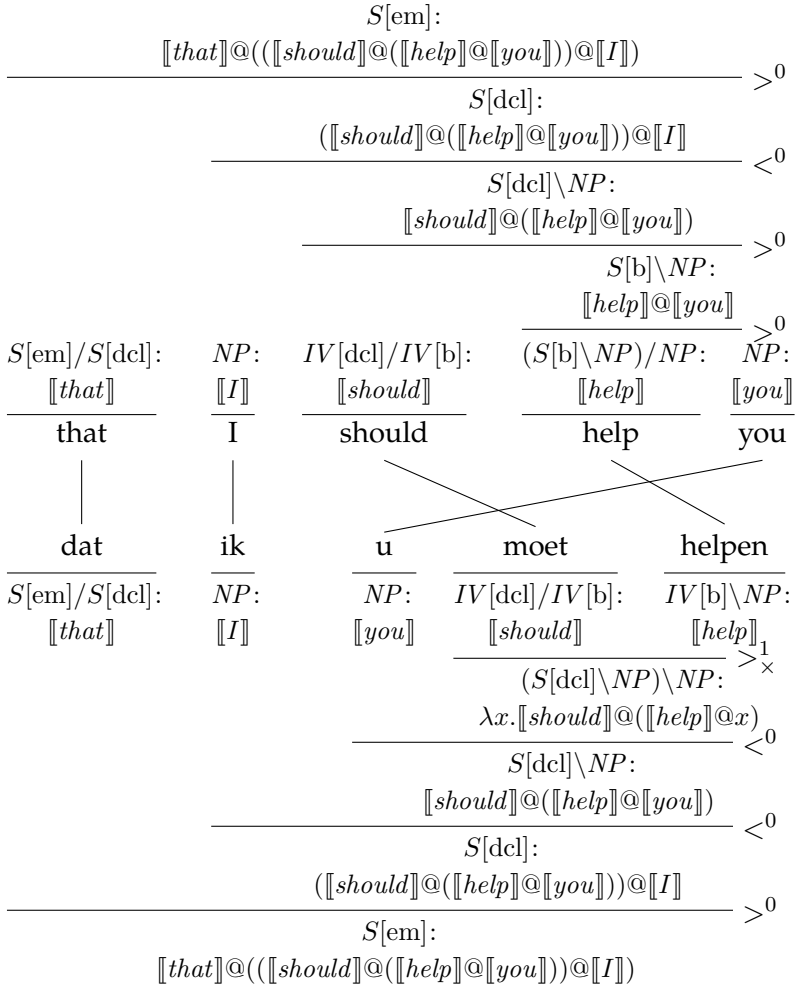


Figure 6.4: Cross-serial word order in Dutch requires the use of crossed composition. $IV[\mathrm{dcl}]$ is an abbreviation for $(S[\mathrm{dcl}]\backslash NP)$. [PMB 98/0884]

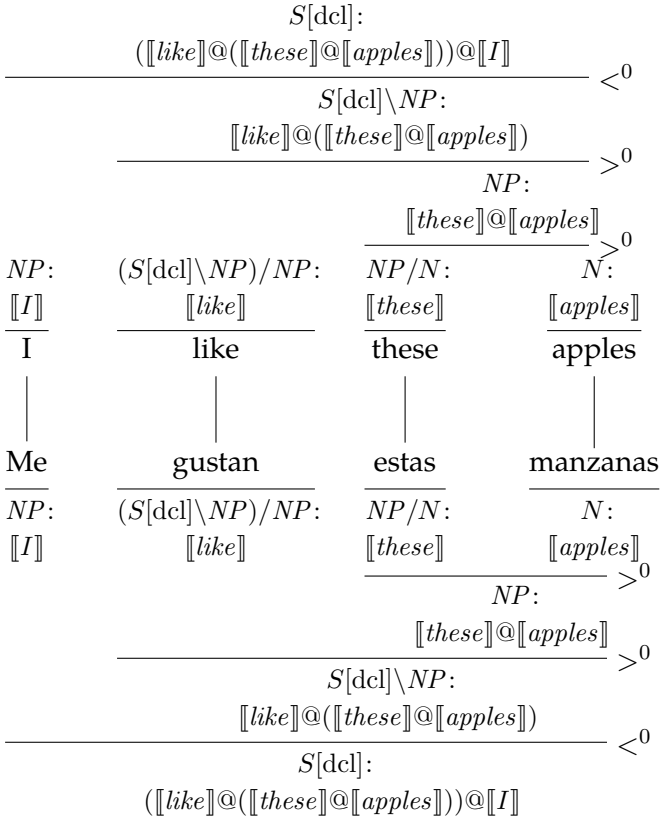


Figure 6.5: Example from Rosetta (1994) illustrating thematic divergence.

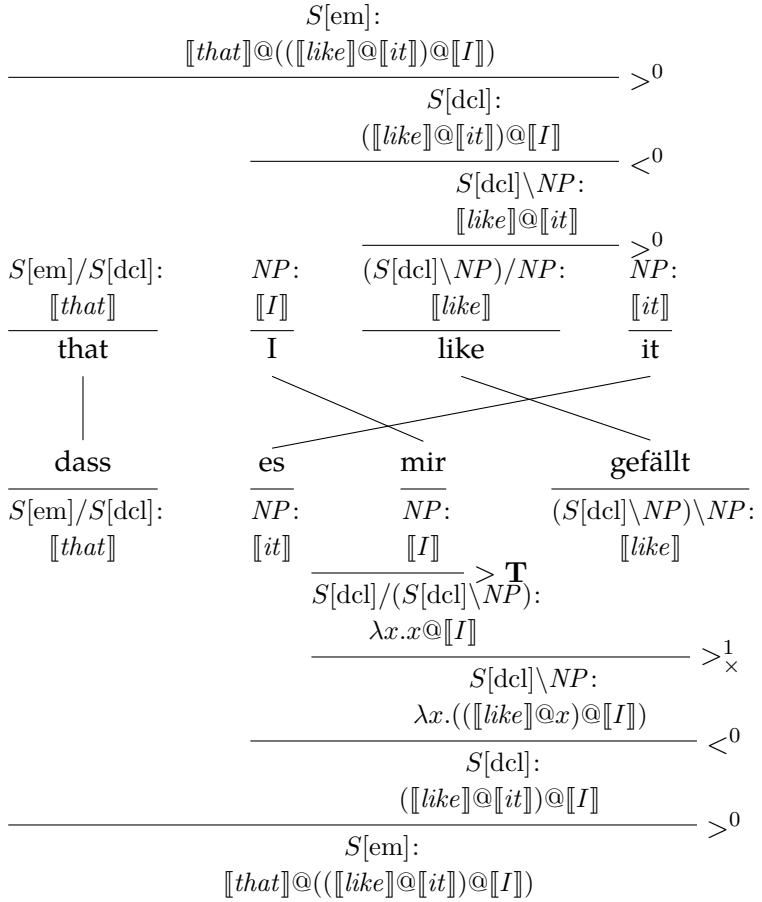


Figure 6.6: Example in which our approach to handling thematic divergence requires the introduction of type-raising and crossed composition.

sary because the subject happens to be of the right of the verb, and the object on its left.

Things are a little more complicated in the English-German example in Figure 6.6, where both arguments, *es* and *mir*, are on the same side of

the verb *gefällt* while the second argument of the verb (according to the English category) is closer to the verb than the first, blocking semantically correct application. In such cases, type-raising in combination with composition allows the arguments to combine with the verb in the order that results in the correct semantics.

6.3 Advanced Derivation Projection Algorithms

6.3.1 ARFCOTRAP: Align, Reorder, Flip, Compose, Transfer and Parse

ARFTRAP still can only handle cases where there is a one-to-one correspondence between source and target language words. For example, if several words in $\mathbf{e}$ are aligned to the same word in $\mathbf{f}$, the TRANSFER step of ARFTRAP fails because it is not specified which of the words of $\mathbf{e}$ should provide the category and interpretation. ARFCOTRAP is the first in a series of refinements gradually relaxing the one-to-one assumption. Its steps are the following:

1. ALIGN: Determine which word in $\mathbf{e}$ corresponds to which word in $\mathbf{f}$.

2. REORDER: Stably sort the words in $\mathbf{e}$ so they appear in the same order as the corresponding words in $\mathbf{f}$.

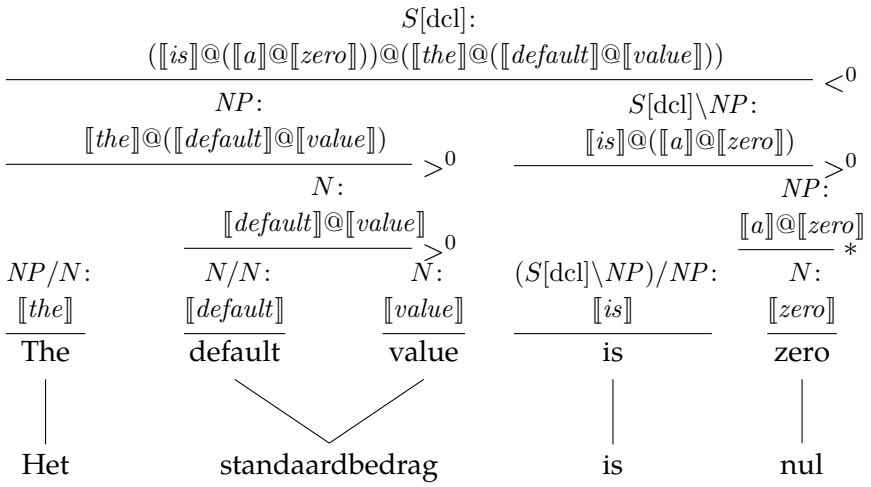
3. FLIP: In the syntactic categories, replace all structure-shared instances of / (\) corresponding to arguments which were moved to the other side of their functor into \ (/).

4. COMPOSE: For every group of $\mathbf{e}$ words that are all aligned to the same $\mathbf{f}$ word, combine them into one constituent using combinatory rules, obeying the normal-form and valid semantic fragment constraints. Replace these words by a single multiword with the same category and interpretation as that constituent.

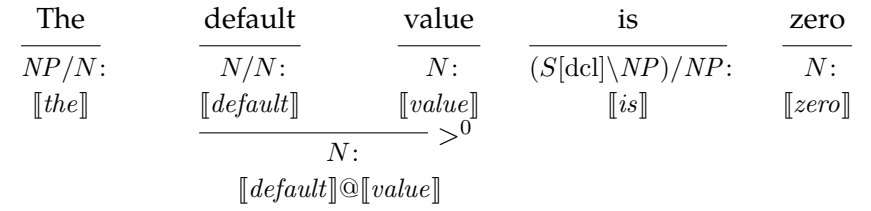
5. TRANSFER: Assign the i-th word in $\mathbf{f}$ the category and interpretation of the i-th word in the modified sentence $\mathbf{e}$.

6. Parse: Apply combinatory rules to obtain a normal-form derivation for **f** that has the same interpretation as that for **e**.

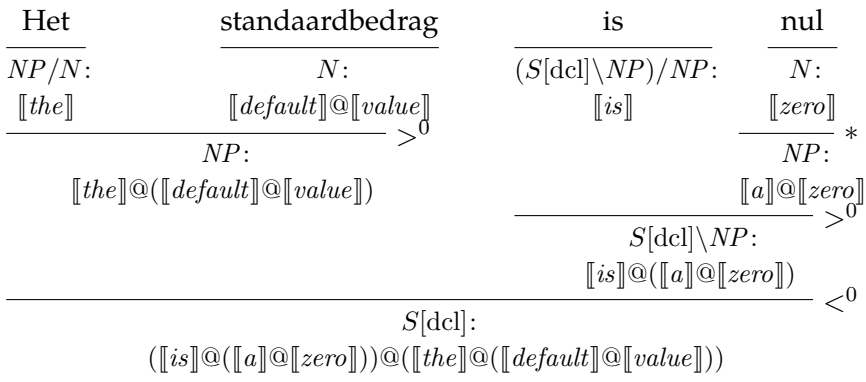
The new Compose step, simply put, puts multiple words together into one, retaining their combined interpretation. Consider the following example [PMB 01/0935]:



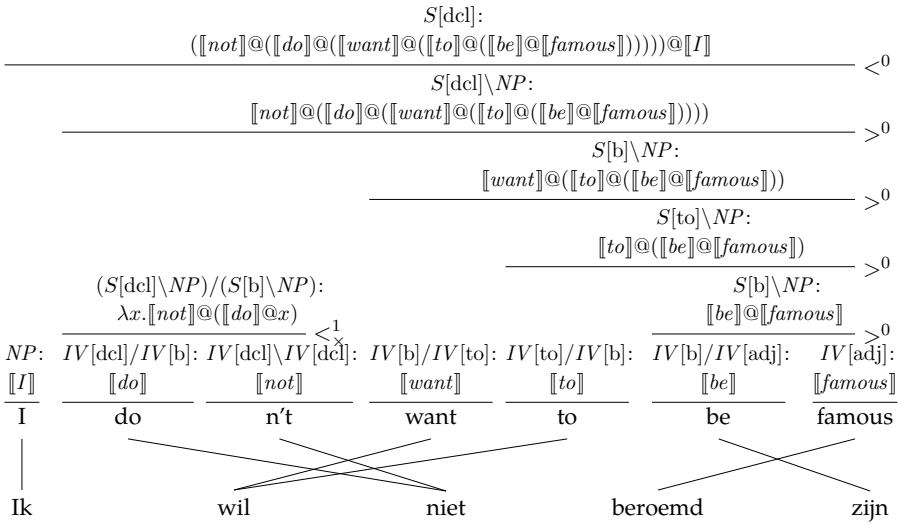
Reordering and flipping are no-ops in this case, but then there remain five words on the English side as opposed to four on the Dutch side. The cause is that *standaardbedrag* is a single (compound) noun whereas the English counterpart *default value* consists of (or is spelled as) two words. In this case, the Compose step applies a single instance of forward application to create a single constituent from them:



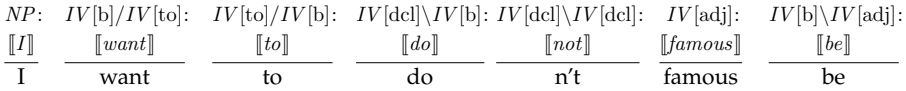
This new constituent is then transferred to *standaardbedrag*, and the PARSE step builds the following derivation:



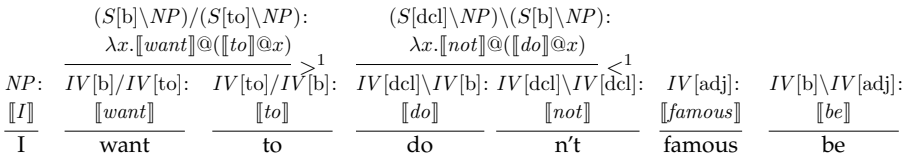
In this relatively straightforward case, the constituent built by the COMPOSE step is identical to a constituent in the original derivation, and its interpretation is a complete subterm of the target interpretation. Neither need be the case. Consider the following example [PMB 01/0942], in which the semantics of two complex English constructions—negation with *do*-support and *want* followed by the infinitival particle *to*—are expressed with fewer words in Dutch:



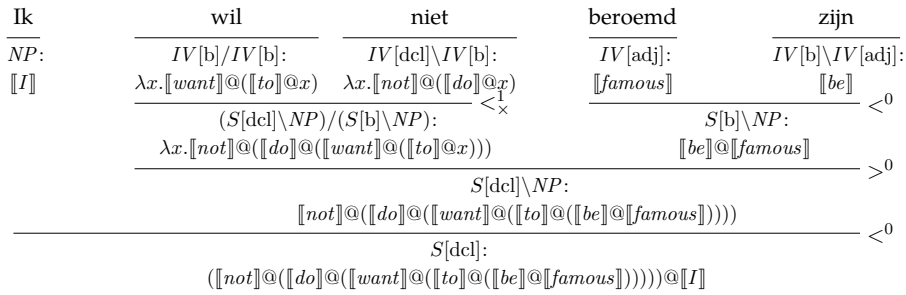
Reordering and flipping gives:



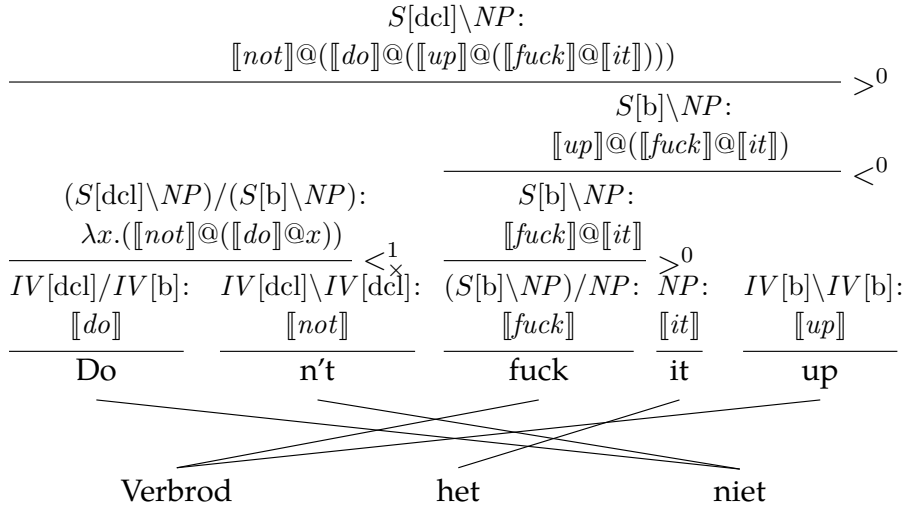
The Compose step then combines *want* and *to* through forward composition and *do* and *n't* through backward composition:



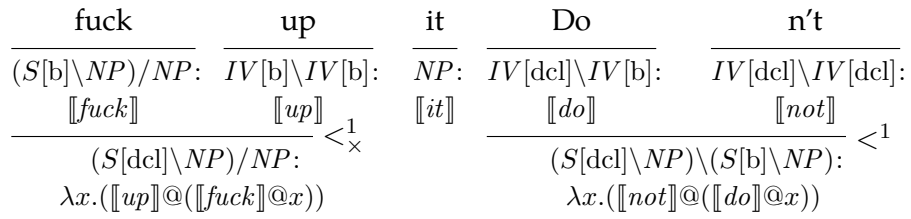
The resulting Dutch derivation is:



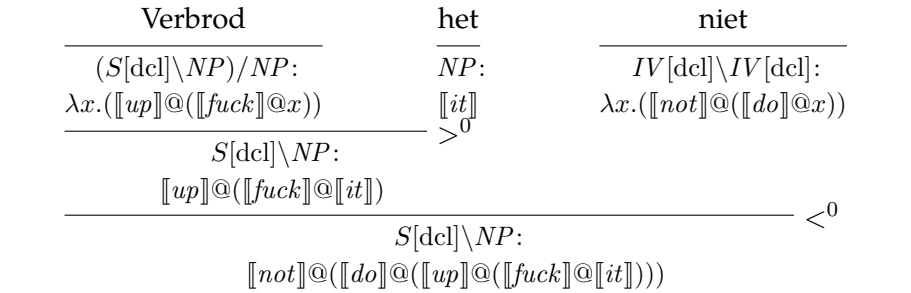
The mechanism can even be extended to *non-contiguous* sets of English words that correspond to a single target-language word, as is often the case with English phrasal verbs. In the PMB, these are given a compositional analysis, with the verb particle acting like a manner adverb. Consider the following example [PMB 88/0916]:



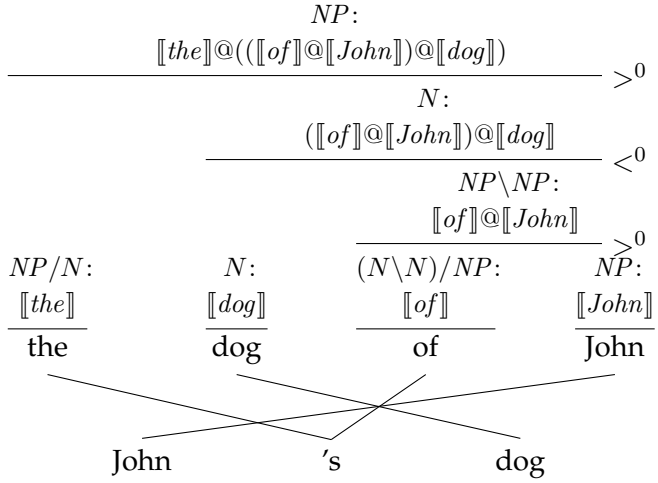
The Reorder step puts *fuck* and *up* together, which enables the Compose step to combine them via backward crossed composition:



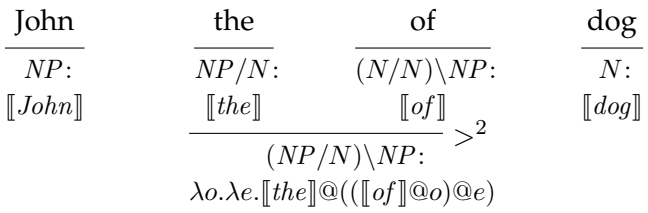
Using the composed categories and interpretations, the resulting (Flemish) Dutch derivation is:



As a further example of the capabilities of Compose, let us consider a somewhat artificial example where two different phrases have the same meaning and also happen to both be English—that is, they are *paraphrases* of each other. One expresses the possession relation using the preposition *of* whereas the other uses the so-called Saxon genitive:

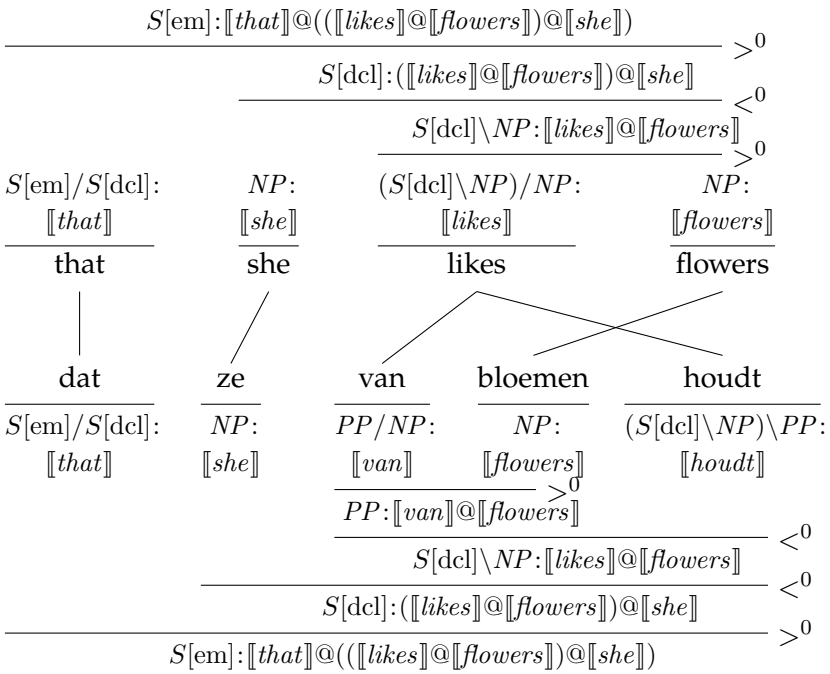


The *'s* token expresses the possession relation like *of*, but it also has the effect of turning *John* into a determiner for *dog*, expressing definiteness like *the*. *'s* should therefore be aligned to both function words. After reordering and flipping, Compose uses generalized composition to produce a new category and interpretation suitable for *'s*:



6.3.2 Arfcostrap: Align, Reorder, Flip, Compose, Split, Transfer and Parse

The opposite case, in which one source-language word corresponds to two or more target-language words, is trickier. Ideally, we would like to split up the category and interpretation in such a way that each target-



where

$$[\![likes]\!] = \lambda do.\lambda su.\lambda mp.(su@\lambda x.(do@\lambda y.(\ \boxed{\begin{array}{|l}\hline e \\ \hline like.v.03(e) \\ Experiencer(e,x) \\ Stimulus(e,y) \\ \hline\end{array}}\ ; mp@e))))$$

$$[\![houdt]\!] = \lambda po.\lambda su.\lambda mp.(su@x.(\ \boxed{\begin{array}{|l}\hline e \\ \hline like.v.03(e) \\ Experiencer(e,x) \\ \hline\end{array}}\ ; ((po@e); (mp@e))))$$

$$[\![van]\!] = \lambda np.\lambda e.(np@\lambda x.\ \boxed{\begin{array}{|l}\hline \\ \hline Stimulus(e,x) \\ \hline\end{array}}\ )$$

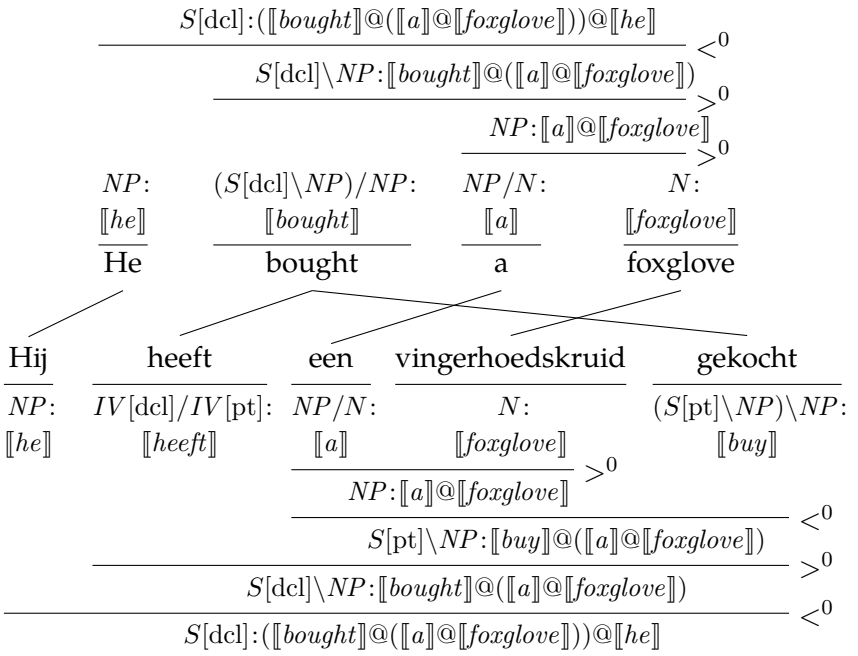
Figure 6.7: A linguistically motivated split of the category and interpretation of *likes*. [PMB 04/0848]

language word receives a linguistically adequate category. We give an example of this in Figure 6.7. The English verb *likes* with a direct object here corresponds to the Dutch verb *houden* with a prepositional object headed by the preposition *van*. Our analysis assigns *van* the syntactic category PP/NP and *houdt* $(S[\mathrm{dcl}]\backslash NP)\backslash PP$, which, when cross-composed, would yield a category identical to that of *likes*, up to slash flipping. For splitting up the semantic category, we look at the λ-DRS expression that our symbol $[\![likes]\!]$ stands for and choose categories so that the category of *van* contributes the semantic role condition for the object and that of *houdt* contributes the remaining conditions.

A similar configuration appears where some grammatical category is expressed synthetically in the source language but analytically in the target language. For example, English frequently expresses past tense using the synthetic simple past form of a verb where the Dutch translation opts for an analytic construction, with a form of *zijn* or *hebben* as an auxiliary, combined with the past participle form of the verb. A semantically adequate analysis would assign the DRS conditions expressing tense to the auxiliary and the rest to the participle, as exemplified in Figure 6.8.

Unfortunately, there is no general procedure to perform such splits mechanically—given a λ-expression h, we could choose from infinitely many f, g such that they combine into h via application or composition (Huet, 1975; Kwiatkowksi et al., 2010), and similarly for the syntactic categories. We cannot hope to find the split linguistically most adequate with a simple mechanical process. However, we can still find a target-language derivation while avoiding splitting. We will give one such solution in this section, and another in the next.

The first solution applies to the case where the group of target-language words aligned to the same source-language word is contiguous. We can then avoid splitting the semantics by treating them as a single *multiword*, or as a *word-with-spaces* in the terminology of Sag et al. (2002). For example, in (2), Dutch uses a definite article to refer to life in general whereas English uses no article. We can sensibly construct a derivation where *het leven* is a word-with-spaces with the same category and interpretation as the English one-word phrase *life*. As another example, in (3) the Dutch *fixed expression* (again borrowing terminology from Sag



where

$$[\![bought]\!] = \lambda do.\lambda su.\lambda mp.(su@\lambda x.(do@\lambda y.(\ \boxed{\begin{array}{|l}\hline e\ t_1\ t_2 \\ \hline buy.v.01(e) \\ Theme(e,y) \\ Agent(e,x) \\ now(t_1) \\ e \subseteq t_1 \\ t_2 < t_1 \\ \hline\end{array}}\ ; mp@e))))$$

$$[\![heeft]\!] = \lambda vp.\lambda su.\lambda mp.((vp@su)@\lambda x.(\ \boxed{\begin{array}{|l}\hline t_1\ t_2 \\ \hline now(t_1) \\ x \subseteq t_1 \\ t_2 < t_1 \\ \hline\end{array}}\ ; (mp@x)))$$

$$[\![buy]\!] = \lambda do.\lambda su.\lambda mp.(su@\lambda x.(do@\lambda y.(\ \boxed{\begin{array}{|l}\hline e\ t_1\ t_2 \\ \hline buy(e) \\ Theme(e,y) \\ Agent(e,x) \\ \hline\end{array}}\ ; mp@e))))$$

Figure 6.8: A linguistically motivated split of the category and interpretation of *bought*. [PMB 91/0888]

et al. (2002)) *op het punt* fulfills the same function as the English word *about* and can be treated as a word-with-spaces with the same category and interpretation. In (4), the adverbial expression *naar beneden* (literally: *to beneath*) is more analytic than the corresponding English adverb *downstairs*, but also contiguous and can therefore be treated as a word-with-spaces.

(2) a. That's *life*.
b. Dat is *het leven*. [PMB 35/0877]

(3) a. He was *about* to go out when the telephone rang.
b. Hij stond *op het punt* weg te gaan toen de telefoon ging. [PMB 00/0917]

(4) a. We heard him come *downstairs*
b. We hoorden hem *naar beneden* komen [PMB 51/0833]

The words-with-spaces solution is implemented by the ARFCOSTRAP algorithm, adding a SPLIT step. This step is so named because it divides up the category and interpretation of a single source-language word among a number of target-language words, which become a multiword.

1. ALIGN: Determine which word in **e** corresponds to which word in **f**.

2. REORDER: Stably sort the words in **e** so they appear in the same order as the corresponding words in **f**.

3. FLIP: In the syntactic categories, replace all structure-shared instances of / (\) corresponding to arguments which were moved to the other side of their functor into \ (/).

4. COMPOSE: For every group of **e** words that are all aligned to the same **f** word, combine them into one constituent using combinatory rules, obeying the normal-form and valid semantic fragment constraints. Replace these words by a single multiword with the same category and interpretation as that constituent.

5. SPLIT: For every group of contiguous **f** words that are all aligned to the same **e** word, combine them into one word-with-spaces.

6. Transfer: Assign the i-th word in **f** the category and interpretation of the i-th word in the modified sentence **e**.

7. Parse: Apply combinatory rules to obtain a normal-form derivation for **f** that has the same interpretation as that for **e**.

6.3.3 Arfcoistrap: Align, Reorder, Flip, Compose, Insert, Split, Transfer, Parse

In cases where two or more *non-contiguous* target-language words correspond to a single English word, as in Figures 6.7 and 6.8, we cannot treat them as words-with-spaces. Another option is to have one word contribute the entire semantics and treat the other(s) as lexically empty, possibly even as a "skipped" word that is not part of the derivation proper. A reanalysis of the sentence pair in Figure 6.7 under this treatment is shown in Figure 6.9.

Note that here we assume a different word alignment than before, one where *van* remains unaligned. This trick can be applied in many cases where function words in the target-language sentence do not have a directly corresponding word in the source-language sentence. This includes auxiliaries such as *heeft* in Figure 6.8, articles such as *het* in (2), reflexive pronouns such as *me* in (5), quantitative *er* (Bech, 1952; Berends et al., 2010) as in (6), existential/expletive *er* (Bech, 1952; Bennis, 1986) as in (7) and complementizers such as *dat* in (8).

(5) a. I feel sick.
b. Ik voel me ziek. [PMB 58/0804]

(6) a. Let's get one.
b. Laten we er een nemen. [PMB 11/0939]

(7) a. I wonder what happened to Paul.
b. Ik vraag me af wat er met Paul gebeurd is. [PMB 35/0784]

(8) a. The manager said it was your fault.
b. De manager zei dat het jouw schuld was. [PMB 45/0931]

Skipping unaligned target-language words is implemented by the Arfcoistrap algorithm. The new Insert step actually *deletes* target-language

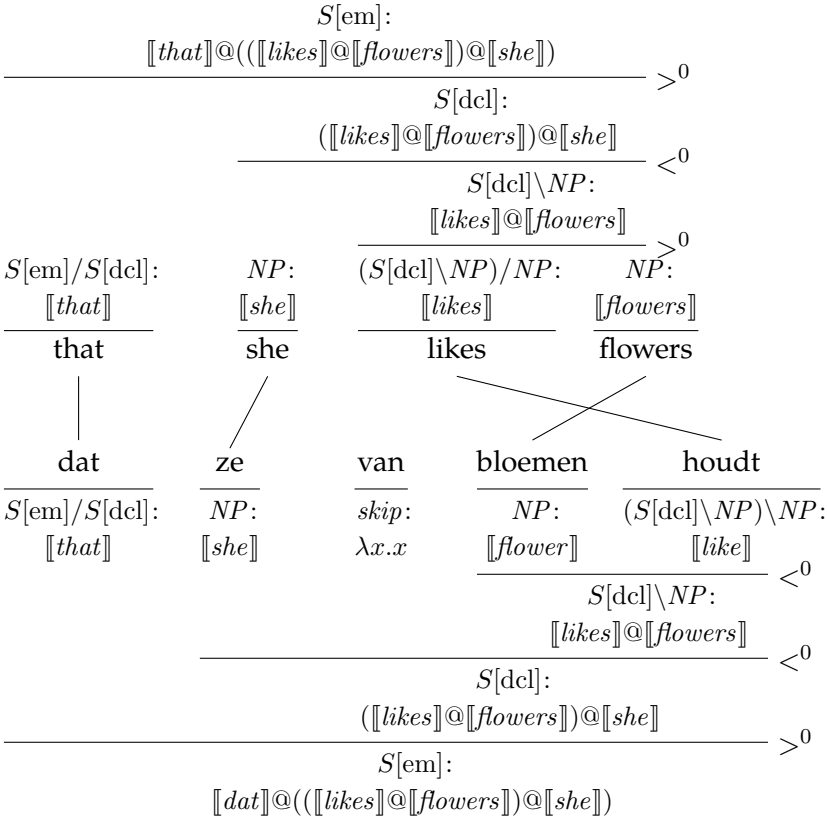


Figure 6.9: Example of skipping: *houdt* receives the (slash-flipped) category of *likes*; *van* is skipped. [PMB 04/0848]

words, but from the perspective of the source-language derivation, it is the insertion of new material into the sentence (but not into the derivation):

1. ALIGN: Determine which word in $\mathbf{e}$ corresponds to which word in $\mathbf{f}$.

2. REORDER: Stably sort the words in $\mathbf{e}$ so they appear in the same order as the corresponding words in $\mathbf{f}$.

3. FLIP: In the syntactic categories, replace all structure-shared instances of $/$ ($\backslash$) corresponding to arguments which were moved to the other side of their functor into $\backslash$ ($/$).

4. COMPOSE: For every group of $\mathbf{e}$ words that are all aligned to the same $\mathbf{f}$ word, combine them into one constituent using combinatory rules, obeying the normal-form and valid semantic fragment constraints. Replace these words by a single multiword with the same category and interpretation as that constituent.

5. INSERT: Delete every unaligned target-language word.

6. SPLIT: For every group of contiguous $\mathbf{f}$ words that are all aligned to the same $\mathbf{e}$ word, combine them into one word-with-spaces.

7. TRANSFER: Assign the i-th word in $\mathbf{f}$ the category and interpretation of the i-th word in the modified sentence $\mathbf{e}$.

8. PARSE: Apply combinatory rules to obtain a normal-form derivation for $\mathbf{f}$ that has the same interpretation as that for $\mathbf{e}$.

In order to train a parser on parallel derivations that contain skipped words, the parser needs to be able to skip certain words and to learn when to do so. We will introduce such a parsing algorithm in Section 7.2.1.

6.4 Unaligned Source-language words

Word alignments as commonly used in machine translation (e.g., Koehn et al. (2003)) are bipartite graphs where the partite sets are the sets of words in a source-language sentence and its target-language translation. Maximal connected subgraphs have either one word of each, in which case they are called 1:1 translation units, one source-language word and several target-language words (1:N translation unit), several source-language words and one target-language word (N:1 translation unit) or only one word (unaligned word).

Arfcoistrap cannot handle 1:N translation units where the target-language words are discontiguous, but it handles the contiguous case through the Split operation. N:1 translation units are handled through the Compose operation. Unaligned target-language words are handled through the Insert Operation. It does not yet handle unaligned source-language words.

In a good alignment, in what situations can we expect unaligned source-language words? One possible example, given by (Koehn, 2010, p. 114), is the word *does* in the English-German sentence pair (9).

(9) a. John does not live here
b. John wohnt hier nicht

As Koehn points out, *does* could be unaligned because it has no clear German equivalent, but it could also be aligned to *wohnt*, because that's what it shares number and tense with, or to *nicht*, because *does* appears because of the negation. We already opted for the third option in a number of Dutch examples above. Consider also (10), where German expresses with a verb what English expresses with a predicative adjective, and thus lacks a copula, and (11), in which the Italian lacks an overt subject pronoun:

(10) a. I am cold
b. Ich friere

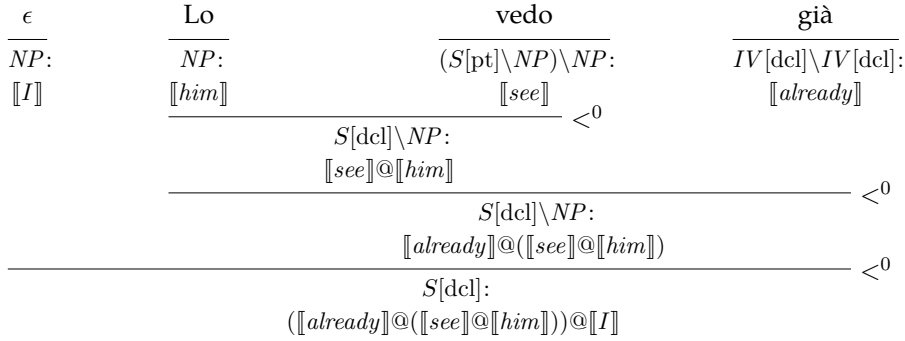
(11) a. I see him already
b. Lo vedo già

The words *am* and *I*, respectively, thus lack a direct equivalent in the translations and could be left unaligned. The question then arises how we make sure that their interpretations (which we treat as atomic and would give as $[\![am]\!]$ and $[\![I]\!]$, respectively) enter into the target-language derivation we construct. On grounds of morphosyntactic features, an argument can be made that *am* should be aligned to *friere* alongside *cold* because *friere* and *am* both contribute the features of first person singular, and present tense. Similarly, one can argue that *I* should be aligned to *vedo* alongside *see* because *vedo* is the first-person singular form of the verb. Indeed, Italian is argued to be a language where pronominal subjects are primarily expressed through verbal affixes (Dryer, 2013b), which would make this choice seem even more fitting. The Compose step of Arfcoistrap would then merge *I* and *see* as follows:

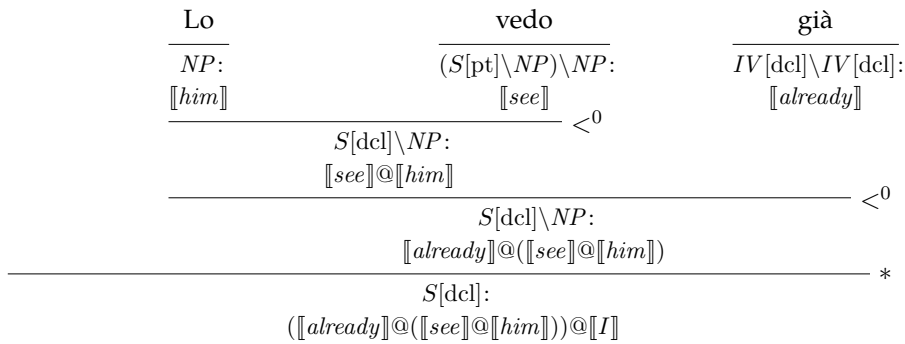
$$\dfrac{\dfrac{\dfrac{\text{him}}{\begin{array}{c}NP\text{:}\\ [\![him]\!]\end{array}} \quad \dfrac{\dfrac{\text{I}}{\begin{array}{c}NP\text{:}\\ [\![I]\!]\end{array}}}{\begin{array}{c}S[\text{dcl}]/(S[\text{dcl}]\backslash NP)\text{:}\\ \lambda f.(f@[\![I]\!])\end{array}}>\mathbf{T} \quad \dfrac{\dfrac{\text{see}}{\begin{array}{c}(S[\text{dcl}]\backslash NP)\backslash NP\text{:}\\ [\![see]\!]\end{array}} \quad \dfrac{\text{already}}{\begin{array}{c}IV[\text{dcl}]\backslash IV[\text{dcl}]\text{:}\\ [\![already]\!]\end{array}}}{\cdots}}{\begin{array}{c}S[\text{dcl}]\backslash NP\text{:}\\ \lambda x.(([\![see]\!]@x)@[\![I]\!])\end{array}}>^1_\times$$

The problem with this is that if we pre-compose subject and verb, verb phrase modification is no longer possible: the category of *already*, transferred to *già*, expects to combine with the verb phrase category $S[\text{dcl}]\backslash NP$ where NP stands for the subject category, but the subject is no longer available. One might consider changing the original English grammar to avoid verb phrase modification and use sentence modification instead.

A different approach to unaligned source-language words would be a Delete step which inserts *empty elements* into the target-language sentence, for example "tokens" that are not part of the written form but which have the categories and interpretations of the unaligned source-language words. For example, (11-b) could then be analyzed as follows:



Alternatively and almost equivalently, the introduction of the semantic material not associated with any target-language word could be accomplished by a unary type-changing rule with a specific interpretation for the introduction of first-person subjects:



In either case, the target-language parser we train will have to learn to predict from contextual features when material should be added. For example, the presence of an Italian first-person singular verb form in the right context can be a clue that an empty first-person singular pronoun is required. Alternatively, a verb phrase being headed by such a verb form could help the parser predict that it should apply the corresponding type-changing rule to it. Similarly, a German verb form associated with the interpretation of an English adjective, as we might have in (10), could give the parser a hint to introduce the semantics of

the English copula via either route. Thereby, existing links, e.g., morphosyntactic ones, between overt target-language words and unaligned source-language words, would still be preserved, albeit not in the CCG lexicon, but in the parsing model.

The introduction of semantic material in parallel derivations without making it part of the interpretation of any target-language word is even more compelling in pairs such as the following, where the French relative clause happens to be reduced, viz. lacking an overt relative pronoun and auxiliary, while the English happens to have both.

(12) a. a survey that was conducted by the EU
b. une enquête menée par l' UE

Here, there is really no good way to associate the semantics of *that* and *was* with any French word, since the exact same reduction could occur, semantically equivalently, in English:

(13) a survey conducted by the EU

In this case, our English derivation would itself use a type-changing rule, for which the following interpretation is adequate. The very same rule could be used in the French interpretation in (12):

(14) $S[\mathrm{pss}]\backslash NP : f \quad \Rightarrow \quad S[\mathrm{dcl}]\backslash NP : [\![that]\!]@([\![be]\!]@f)$

In conclusion, we find that either type-changing rules or empty elements are the best way to deal with unaligned source-language words within our framework. On one hand, type-changing rules seem simpler because they are not directional whereas for empty elements a decision would have to be made as to where to insert them, left or right of the constituent whose semantics is to be changed. On the other hand, empty elements may generalize better because they can behave syntactically the same as overt elements—for example, an empty Italian subject pronoun can behave like a non-pronominal subject. We leave these questions to future work and for now contend ourselves with the limited capabilities of ARFCOISTRAP.

6.5 Handling of Translation Divergences

A sentence and its translation often express the same thing in a different way. Not only are different words and different phrase orders used, often the sentences are also structurally different. In the Machine Translation literature, such phenomena are known as *translation divergences*. We have used a number of instances of divergences above to motivate the algorithmic steps of Compose, Insert, Split and the hypothetical Delete. But are these steps sufficient to handle all—or even many—of the translation divergences we can expect to find in the data?

In order to answer this question, we turn to the well-known survey of translation examples by Lindop and Tsujii (1991), later reorganized by Dorr (1993) into a hierarchy that she argues covers the entire range of possible translation divergences. We will sketch for each of Dorr's divergence types whether and how Arfcoistrap handles it, and how well we can expect the resulting lexical items to generalize. We will use a representative subset of the original examples, sometimes reordered to have English, our source language, first.

6.5.1 Thematic Divergences

Thematic divergences occur when a verb assigns the same semantic roles different syntactic roles than the translation, as in these examples by Lindop and Tsujii:

(15) a. I like the car
b. Mir gefällt der Wagen

(16) a. John misses Mary
b. Mary manque à John

(17) a. He lacks something
b. Ihm fehlt etwas

Our algorithm has no notion of syntactic roles. In each example, it will build the parallel derivations in such a way that the interpretations of the arguments are taken by the verb interpretation in the same order as in the source language, using combinatory rules, including type-

raising, as needed (cf. Section 6.2.3). This certainly works for the examples presented, and also for different argument orders as we have seen in Figure 6.6.

The resulting derivations are a bit odd since we preserve the arbitrary semantic argument order of English and change the syntactically motivated syntactic argument order. For example, in a normal analysis of (16-a), the interpretation of *miss* takes the Stimulus as first argument and the Experiencer as second just because this matches the canonical object-subject order of combining. In preserving the semantic but not syntactic argument order, we obtain a derivation for (16-b) in which the verb takes the subject *Mary* before the object. If one knows that *Mary* is the subject here, one might change the verb category and interpretation correspondingly so as to obtain more uniform categories and derivations, better generalization for statistical parsing and a grammar that supports binding theory along the lines of (Steedman, 2001, Section 4.3.1).

For German and other languages with freer word order, treating argument types and their order with the appropriate generalizations is more intricate and may require additional formal machinery. Treatments at different levels of departure from classical CCG have been proposed, for example, in Hockenmaier (2006); Steedman and Baldridge (2011); Hoffman (1995).

6.5.2 Structural Divergences

Like *thematic divergences*, the term *structural divergences* describes divergences in argument structures, primarily of verbs. Whereas the former term describes syntactic positions and case assigned to arguments, the latter describes changes in the *internal* structure of arguments, e.g., expression as an NP vs. as a PP (18), (19), or as a bare adjective vs. as a VP with a copula (20):

(18) a. He aims the gun at him
b. Er zielt auf ihn mit dem Gewehr

(19) a. Der Student beantwortet die Frage
b. L'étudiant répond à la question

(20) a. He seems ill
b. Hij schijnt ziek te zijn

Our algorithm's answer to such divergences is to treat "extra" words in the target language, like *mit*, *à* or *te zijn*, as skipped, as described in Section 6.3.3. The opposite case, where the source language contains extra (function) words, would require the use of empty elements or type-changing rules, as discussed in Section 6.4. A parser expected to create similar representations from unseen text will have to predict when to skip, insert or type-change, and how the choice of lexical categories and interpretations in the context is affected.

6.5.3 Categorial Divergences

While structural divergences concern the expression of *arguments* as phrases of different types, *categorial divergences* concern the expression of *functors*, such as verb phrases or modifiers, as phrases of different types.

(21) a. Postwar
b. Nach dem Krieg

(22) a. He resides in Amsterdam
b. Hij is in Amsterdam woonachtig

(23) a. It suffices
b. Het is voldoende

(24) a. Hopefully
b. On espère

(25) a. John is fond of music
b. John aime la musique

When a single-word source-language modifier corresponds to a complex target-language phrase, as in (21) or (24), the category our algorithm assigns to the target-language phrase is usually perfectly adequate. For example, *postwar* (as an attributive adjective) would have category N/N, *nach dem Krieg* would get $N\backslash N$, and this is exactly the right category for noun modifiers, regardless of their internal structure.

The same argument applies in the opposite direction, when a source-language phrase is composed to provide the category for a single target-language word.

(22) and (23) are examples of English verbs corresponding to Dutch adjectives, where the Dutch uses a copula. The linguistically most adequate solution would be to "split" the category of the English verb into $(S[\mathrm{dcl}]\backslash NP)/(S[\mathrm{adj}]\backslash NP)$ for the copula and $S[\mathrm{adj}]\backslash NP$ for the adjective, similar to Figure 6.8. However, this is beyond the capabilities of our algorithm. What it can do is treat the copula as skipped and assign the Dutch adjective the verbal category. While this works, it can lead to a noisy target-language lexicon where there is no clear link between the part-of-speech of a word and its category, and its interpretation may express features such as tense that are not actually marked on the word, only on the aligned English one in some examples. In the opposite direction, this problem does not occur, e.g., composing *is fond of* yields the category $(S[\mathrm{dcl}]\backslash NP)/NP$, which is exactly right for *aime*.

6.5.4 Head Switching (Promotional and Demotional Divergences)

Head switching (subdivided by Dorr into promotional and demotional divergences) is a type of divergence whereby in one sentence, a semantic contribution is made by an adjunct to a phrase (e.g., an adverb to a verb phrase), and in the other sentence, it is made by the head of an additional phrase embedding the corresponding phrase (e.g., an additional verb phrase embedding the corresponding verb phrase). The examples are:

(26) a. He happens to be ill
b. Hij is toevallig ziek

(27) a. Jean will probably come
b. Il est probable que Jean viendra

(28) a. The baby just fell
b. Le bébé vient de tomber

(29) a. An attempted murder

b. Une tentative de meurtre

(30) a. He likes reading
b. Er liest gern

The beauty of CCG in this case is that both structures receive structurally very similar analyses: whether a phrase α is modified by an adjunct β or embedded as an argument of another head β, in both cases CCG will analyze β as a functor taking α as an argument. For example, the modifier *gern* in (30-b) takes the VP *liest* as argument and returns another VP, and the same is true for the matrix verb *likes* and the VP *reading* in (30-a). Our algorithm thus produces derivations with the same interpretation for *likes* and *gern*. Similarly, *happens to* (after composing) can provide the category and interpretation for the adverb *toevallig*, and *Il est probable que*, *vient de* and *tentative de* can all be treated as words-with-spaces with categories and interpretations from the corresponding English modifiers.

Thus, under the CCG analysis, head switching is almost a non-divergence. The remaining divergence is in the morphosyntactic features: modifiers such as *gern* map VPs of any type to VPs of the same type whereas VP arguments usually have types like $S[\mathrm{ng}]\backslash NP$ or $S[\mathrm{b}]\backslash NP$. In (30), we would obtain the category $(S[\mathrm{dcl}]\backslash NP)\backslash(S[\mathrm{ng}]\backslash NP)$ for *gern*. To a syntactician familiar with German, this is simply wrong: the adverb *gern* does not require the verb it modifies to be in present progressive form (ng) nor does it necessarily occur in a finite or declarative (dcl) VP, nor is it even clear that this categorization of VPs makes sense for German. Similar considerations apply to the other examples. Thus again, future work on more informed treatment of target-language syntax may lead to grammars that generalize better.

6.5.5 Conflational Divergences

Conflational divergences occur when one sentence expresses in one or more words what another expresses in multiple words. Simple vs. compound nouns are among the most straightforward examples, easy to deal with by composing in one direction, and words-with-spaces in the other direction:

(31) a. Piscina
b. Swimming pool

There are also conflational differences involving a target-language verb whose source-language translation consists of a verb plus additional elements such as VP modifiers:

(32) a. Get up early
b. Madrugar

(33) a. John called up Mary
b. John a appelé Mary

(34) a. See again
b. Revoir

(35) a. Go out again
b. Ressortir

(36) a. Know how
b. Savoir

Our algorithm normally deals with these cases by combining the category and interpretation of the source-language words through the COMPOSE operation.

The opposite case is source language verbs expressed in the translation as a verb with some modifier:

(37) a. Miss
b. Sentir a falta

(38) a. Float
b. Aller en flottant

Here, the target-language words can be treated as a word-with-spaces, but only if they all appear contiguously, which is often not the case, for example:

(39) a. We floated east
b. Nous allions à l'est en flottant

Here, our algorithm as it stands is unable to find an adequate target-language derivation. A more intelligent SPLIT operation as sketched in Section 6.3.2 would be required.

The following example is like (38) in that part of the semantic content of the English verb is expressed by a target-language VP modifier, but adds the twist that conversely, the content of the English modifier is expressed by the target-language verb.

(40) a. Walk across
b. Traverser à pied

If we align the words mainly by semantic content, in principle we could give *à pied* a verb category and *traverser* a VP modifier category, although this would lead to odd lexical items very specific to this construction. The corresponding entry for *walk*, for example, could not be used in a simple VP like *walk happily*. Here, a finer-grained semantic analysis of lexical items would be required.

6.5.6 Lexical Divergences

It is a basic fact of translation that a good translation of a single word in some context is not necessarily a good translation of the same word in another context. Words are homonymous and polysemous, and the mapping of words to senses is not isomorphic for any two languages. Thus, the choice of target-language word in machine translation is a hard problem, and so is the choice of lexical interpretation in semantic parsing. Here is a basic example where *free* has two different senses (and two different German translations):

(41) a. The suspect is free again
b. Der Verdächtige ist wieder frei

(42) a. The lunch is free
b. Das Mittagessen ist gratis

Sometimes the choice of interpretation and translation for some word is informed by other divergence types, e.g., in the following example of categorial divergence (Dorr, 1993, p. 264–265), we might produce a

derivation in which the German *habe* (literally: *have*) is interpreted like the English *am*.

(43) a. I am hungry
b. Ich habe Hunger

Dorr calls this a lexical "divergence" apparently because *habe* is not a "literal" translation for *am* and only adequate due to the presence of that categorial divergence. Similarly, she presents *like* as a lexically divergent translation for *gustar* (literally: *please*) in the context of a thematic divergence and *dar* (literally: *give*) for *stab* in the context of a conflational divergence. But since the problem of interpretation (and, in machine translation, lexical choice) *always* arises, whether or not due to other divergences of other types, it is questionable whether it is worth distinguishing such cases as another divergence type. Our algorithm does not need any special mechanisms to deal with lexical divergence because lexical items with different, and sometimes "non-literal", word-sense mappings, fall out from it.

A related issue is that of word alignments. Until now, we have treated these as given—an assumption we will be doing away with in the next chapter.

6.6 Conclusions

In this chapter, we have investigated the problem of projecting a semantic parse from a sentence to its translation automatically, given a suitable word alignment. For doing this, it is advantageous to have semantic parses based on a grammar formalism which supports a high degree of parallelism in the analysis of both languages, as this reduces the number of choices to make (e.g., what is a constituent, what are the labels of constituents). We find that CCG, with its small, rather language-independent set of basic categories, and its flexible notion of constituency, fits this bill nicely. The algorithm developed in this chapter, projecting CCG derivations from a source-language sentence to its target-language translation, demonstrates this.

We have shown that the algorithm even works with translations that exhibit thematic, structural, categorial, head-switching and conflational divergences. There are, however, limitations: thematic divergences can lead to awkward analyses that miss appropriate generalizations for the target-language grammar. Structural divergences involving source-language function words with no target-language equivalent are not handled adequately yet and would require insertion of empty elements or special type-changing rules for the target language. For categorial and conflational divergences where the target language is more analytic than the source language, the algorithm lacks a way of splitting categories and interpretations in an appropriate way and therefore creates noisy lexical constituents that generalize poorly. Morphosyntactic features are language-specific and also give rise to poorly generalizing lexical entries.

The purpose of projecting CCG derivations is cross-lingual training of semantic parsers. In order to put this into practice, we will have to deal with the above limitations, as well as noisy word alignments and non-literal (e.g., informative, loose and idiomatic) translations. We will tackle this challenge in the next chapter.

Chapter 7

A CCG Approach to Cross-lingual Semantic Parsing

7.1 Introduction

This chapter aims to show that cross-lingual learning can help create semantic parsers for new languages with minimal human intervention. We present a method that takes an existing (source-language) semantic parser and parallel data and learns a semantic parser for the target language.

Tackling the problem of learning a semantic parser in this way may seem roundabout, but it has potential advantages. Explicit training data for a complex task like semantic parsing is scarce and costly to obtain. Parallel data is more readily available. If an existing semantic parser for one language is available, meaning representations can be obtained cheaply and in big numbers. Although the output of this system will not have the quality of human annotations, it may be able to make up for quality by quantity.

We have seen in Chapter 6 that CCG derivations can in many cases be projected automatically from source-language sentences to their target-language translations. We make use of this to obtain target-langu-

age derivations and train a statistical parser on them. However, the derivation projection algorithm we developed so far is not yet practical because it relies on perfect, unambiguous word alignments, which are difficult to obtain. In Section 7.2, we describe our method, making use of a more robust solution. In a first step, *category projection*, target-language words are assigned categories and interpretations. In a second step, *derivation projection* proper, target-language derivations are built. The third step of our method, *parser learning*, is then concerned with creating the actual semantic parser for the target language.

Our method is in principle applicable to all parsers producing interpreted CCG derivations. It is independent of the concrete meaning representation formalism used, as long as meaning representations are assembled in the standard CCG way (cf. Chapter 2) using the lambda calculus.

We evaluate our method in Section 7.3 by applying it to English as source language, Dutch as target language and Discourse Representation Theory as meaning representation formalism, and measuring the performance of the obtained Dutch semantic parser.

7.1.1 Related Work

In Chapter 3, we reviewed the literature on semantic parsing. In the present chapter, we present a method inspired in particular by Zettlemoyer and Collins (2005) in that we (over)generate candidate CCG lexical entries (through manually specified templates in their case and cross-lingually via word alignments in our case) and let a parser training algorithm figure out which ones to use and in which contexts. However, that approach was only applied to narrow-coverage domains. For broad-coverage parsing, as we have seen, until very recently, all work relied on a manually crafted semantic lexicon for the language to be parsed. We do not wish to assume this for the target language. All broad-coverage semantic parser learning work that we are aware of also relies on an existing syntactic parser to constrain the search space or to provide features. We do not wish to assume this either, because our method should be applicable to under-resourced languages. Our approach does assume an external syntactic and semantic parser for the *source* language,

but not for the *target* language. The approach perhaps most similar to ours is that of Artzi et al. (2015) because it also uses CCG. It cannot directly be applied in the cross-lingual setting, though, because it employs a hand-crafted, language-specific seed lexicon. Unlike theirs, our method currently still lacks any treatment of non-compositional phenomena such as anaphora resolution.

Cross-lingual learning has previously been applied to different natural-language processing tasks, notably part-of-speech tagging and dependency parsing. Three families of approaches can be distinguished. In **annotation projection**, existing annotations of source-language text are automatically projected to target-language translations in a parallel corpus; the result is used to train a target-language system (Hwa et al., 2005; Tiedemann, 2014). In **model transfer**, parsers for different languages share some of their model parameters, thereby using information from annotations in multiple languages at the same time (Zeman and Resnik, 2008; Ganchev et al., 2009; McDonald et al., 2011; Naseem et al., 2012; Täckström et al., 2013). The **translation approach** pioneered by Tiedemann et al. (2014) is similar to annotation projection, but instead of relying on existing translations, it automatically translates the data and synchronously projects annotations to the translation result. Our approach falls within the annotation projection family, with the new challenge that entire CCG derivations with logical interpretations need to be transferred.

7.1.2 The Problem of Noisy Word Alignments

In Chapter 6, we developed an algorithm for projecting source-language CCG derivations to target-language translations—but we assumed perfect, unambiguous word alignments. Such alignments could perhaps be obtained through costly, painstaking manual annotation, but that would defeat our goal of adapting semantic parsers with little to no human intervention. So we have to make do with automatically induced word alignments.

Automatically induced word alignments in parallel corpora are historically a byproduct of word-based models for statistical machine translation. The most widely used such model is IBM Model 4 (Brown et al.,

1993). It uses probability distributions for how many target-language words each source-language word translates to (fertility), which words these are (translation probabilities) and how far their position in the target sentence is removed from the position of the source-language word in the source sentence (reordering or distortion probabilities). These distributions are usually estimated over a sentence-aligned (but not otherwise labeled) parallel corpus using the EM (expectation maximization) algorithm. Once they are estimated, the most probable alignment (the Viterbi alignment) for any sentence pair can be found. It represents the model's best guess as to which words in the target sentence are translations of which word in the source sentence. As such, it can be used to make a bilingual dictionary: each translation unit in a Viterbi alignment indicates a probable translation for the source word. This is done, e.g., in phrase-based statistical machine translation (Koehn et al., 2003), and we will use it to induce CCG lexical entries for the target language.

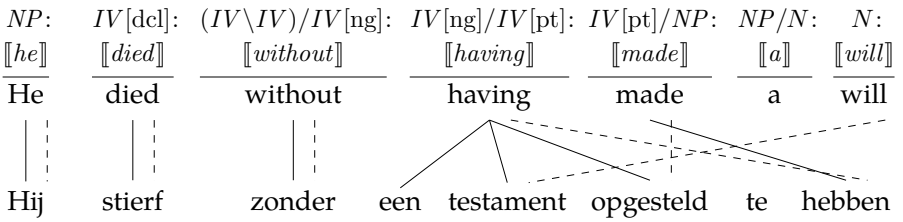


Figure 7.1: Two different word alignments, only partially correct.

Automatically induced word alignments are not perfect. Consider, for example, the sentence pair in Figure 7.1. Solid lines show the Viterbi alignment resulting from training a Model 4 using the GIZA++ implementation (Och and Ney, 2003) with default settings on our training data, with English as source language and Dutch as target language. Some translation units are correct (*Hij*/*He*, *died*/*stierf*, *without*/*zonder*), other translation units are incorrect (e.g., *having* is aligned to *een testament opgesteld*, which means *made a will*), and some are missing. For example, no Dutch word is aligned to the important word *will*, so the Arfcoistrap algorithm from Chapter 6 could not even create an incorrect derivation with the correct interpretation.

We can consider alternative alignments. For example, the dashed lines represent the 3rd-best alignment according to GIZA++ run in the reverse direction, with Dutch as source and English as target. Here, *testament* is aligned correctly to *will*, although we are still missing a correct alignment for *een*. It seems we should not rely on one alignment alone to get all lexical entries needed for making a target-language derivation, but need to gather translation units from multiple alternative alignments.

With these ideas, let us move on to the detailed description of our method.

7.2 Method

We start with a parallel corpus of sentence pairs whose source-language part has been annotated with semantic CCG derivations by the source-language system. We use this annotation in two ways: first, to induce a target-language lexicon in a first step called **category projection**. Secondly, we use it as a form of indirect supervision: we assume that the source-language system works mostly correctly, and that if two sentences are translations of each other, they should have the same interpretation—thus we can train the target-language parser to produce the same interpretations as the source-language parser. To this end, we try to find target-language derivations resulting in the same interpretations as the source-language ones, based on the target-language candidate lexical entries found in category projection. We call this second step **derivation projection**. The derivations thus found are used to train a statistical parsing model for the target language. We call this third step **parser learning**.

All three steps make use of a shift-reduce CCG parsing system which we describe in Section 7.2.1 before moving on to describe the three steps in detail: category projection in Section 7.2.2, derivation projection in Section 7.2.3 and parser learning in Section 7.2.4.

7.2.1 An Extended Shift-reduce Transition System for CCG Parsing

In recent years, transition-based techniques which process linguistic input incrementally have all but taken over syntactic parsing. Their chief advantage is that they work in linear time (Nivre, 2003)—as opposed to chart parsers which work in cubic time—while empirically boasting the same or better accuracy (Zhang and Nivre, 2011; Bohnet and Nivre, 2012; Weiss et al., 2015). They also form a more cognitively plausible model of human sentence processing since humans are known to process linguistic input incrementally and with limited memory (Keller, 2010). Transition-based parsers have also successfully been applied to semantic parsing (Zhao and Huang, 2015; Berant and Liang, 2015), and so does this work.

The transition-based parsing algorithm used is very similar to the syntactic shift-reduce CCG parser of Zhang and Clark (2011), but extended with the capabilities to skip semantically empty words and to use multiwords. It is a transition-based parser, i.e., it works by exploring a search tree of *parser items*. Like most transition-based parsers, it is also *incremental*, i.e., each parser item represents a state of the parser where a prefix of the sentence has already been processed, and the rest has not yet been processed, and for each item, the processed prefix is longer or equal compared to its predecessor. Items are either *unfinished* or *finished*. A finished item corresponds to a parse found by the parser. It typically contains a single parse tree spanning the entire sentence but can also contain a fragmentary parse consisting of multiple trees, each spanning a substring of the sentence.

In our parser, items take the form $\langle S, Q, F \rangle$. S is a list called the *stack*. It contains trees (CCG derivations) covering substrings of the processed prefix. Q is a list called the queue. It represents the suffix of the sentence that has not yet been processed. $F \in \{0, 1\}$ indicates whether or not the item is finished.

In Chapter 2, we defined a CCG as a triple $\langle L, U, B \rangle$ where L is the lexicon and U and B are the sets of unary and binary rule instances. Given such a triple and a sentence $w_1 w_2 \dots w_n$, the *initial item* is $\langle \langle \rangle, \langle L_1, L_2, \dots, L_n \rangle, 0 \rangle$ where $L_i = \{ \frac{w_i w_{i+1} \dots w_j}{C:I} \mid w_i w_{i+1} \dots w_j := C{:}I \in L, i \leq$

$j \leq n\}$. That is, the stack is initially empty and the queue is initialized to contain a set for each position in the sentence, containing all the lexical constituents, including multiwords, that could start at this position.

We define the function $succ_{U,B}$ which maps each item to the set of its immediate successors in the search tree. Each successor is generated by one of six types of parser *action*:

- SHIFT-C-I. This action selects a lexical entry with category C and interpretation I from the candidates at the beginning of the queue of an unfinished item, places it on top of the stack and removes the prefix of the appropriate length from the queue. Formally: for every item $i = \langle\langle s_1, s_2, \ldots, s_m\rangle, \langle q_k, q_{k+1}, \ldots, q_n\rangle, 0\rangle$ and every $\frac{w_k, w_{k+1}, \ldots, w_p}{C:I} \in q_k$, $\langle\langle s_1, s_2, \ldots, s_m, \frac{w_k, w_{k+1}, \ldots, w_p}{C:I}\rangle, \langle q_{p+1}, \ldots, q_n\rangle, 0\rangle \in succ_{U,B}(i)$.

- SKIP. To apply a SHIFT action is to select between the different possible lexical entries for the beginning of the queue, including the choice between single words and multiwords and between different possible categories and interpretations, as specified by the lexicon. If available, the SHIFT action can also select the *skip* category. A constituent with this category needs to be removed from the top of the stack before further actions can be applied because no binary or unary rules accept it as input. This is the sole purpose of the SKIP action. Formally: for every item $i = \langle\langle s_1, s_2, \ldots, s_{m-1}, \frac{w_k}{skip:\lambda x.x}\rangle, Q, 0\rangle$, $\langle\langle s_1, s_2, \ldots, s_{m-1}\rangle, Q, 0\rangle \in succ_{U,B}(i)$.

- BINARY-d-C. This action looks at the top two constituents on the stack of an unfinished item, and if that item was *not* produced by a SKIP action and there is a binary rule in the grammar matching their categories, it applies the binary rule, replacing the two constituents with a single combined constituent. Formally: for every item $i = \langle\langle s_1, s_2, \ldots, \frac{\cdots}{C_1:I_1}, \frac{\cdots}{C_2:I_2}\rangle, Q, 0\rangle$ that was not produced by a SKIP action and every rule instance $C_1 : I_1\ C_2 : I_2 \Rightarrow C : I \in B$ which is an instance of the rule schema R, $\langle\langle s_1\ , s_2, \ldots, \frac{\frac{\cdots}{C_1:I_1}\ \frac{\cdots}{C_2:I_2}}{C:I}R\rangle, Q, 0\rangle \in succ_{U,B}(i)$. $d \in \{\text{L}, \text{R}\}$ indicates whether the left or right

input constituent has the lexical head, serving to guide the parser's choice of lexical features for statistical models. The input constituent with the lexical head is the left one for forward rules and the right one for backward rules, except when that input constituent is a modifier, in which case the other input constituent has the lexical head.

- UNARY-C. This action looks at the top constituent on the stack of an unfinished item. If that item was *not* produced by a SKIP action and there is a unary rule in the grammar matching its category, it applies the unary rule, replacing the constituents with the resulting constituent. Formally: for every item $i = \langle\langle s_1, s_2, \ldots, \frac{\cdots}{C_1:I_1}\rangle, Q, 0\rangle$ not produced by a SKIP action and every $C_1 : I_1 \Rightarrow C : I \in U$ which is an instance of the rule schema R, $\langle\langle s_1, s_2, \ldots, \frac{\frac{\cdots}{C_1:I_1}R}{C:I}\rangle, Q, 0\rangle \in succ_{U,B}(i)$.

- FINISH. This action can take any unfinished item with an empty queue, turn it into a finished item and thereby declare it to be parser output. Formally: for every item $i = \langle S, \langle\rangle, 0\rangle$, $\langle S, \langle\rangle, 1\rangle \in succ_{U,B}(i)$.

- IDLE. This action produces an identical item as successor of each finished item. It was introduced by (Zhu et al., 2013) because it makes it easier to compare parser outputs with different distances from the search tree root in scoring. Being able to retrieve all parser outputs at the same search tree depth also simplifies the formulation of the parsing algorithm. Formally: For every item $i = \langle S, Q, 1\rangle$, $\langle S, Q, 1\rangle \in succ_{U,B}(i)$.

Nothing else is in $succ_{U,B}(i)$ for any item i.

Figure 7.2 shows part of the search tree for the sentence *het leven zuigt* (*life sucks*) under some hypothetical lexicon that we might derive for Dutch in cross-lingual semantic parsing. The initial queue contains separate lexical entries for the words *het* and *leven* as well as a combined lexical entry treating them as a word-with-spaces. This results in two parses with the same interpretation. Items at the same search tree depth

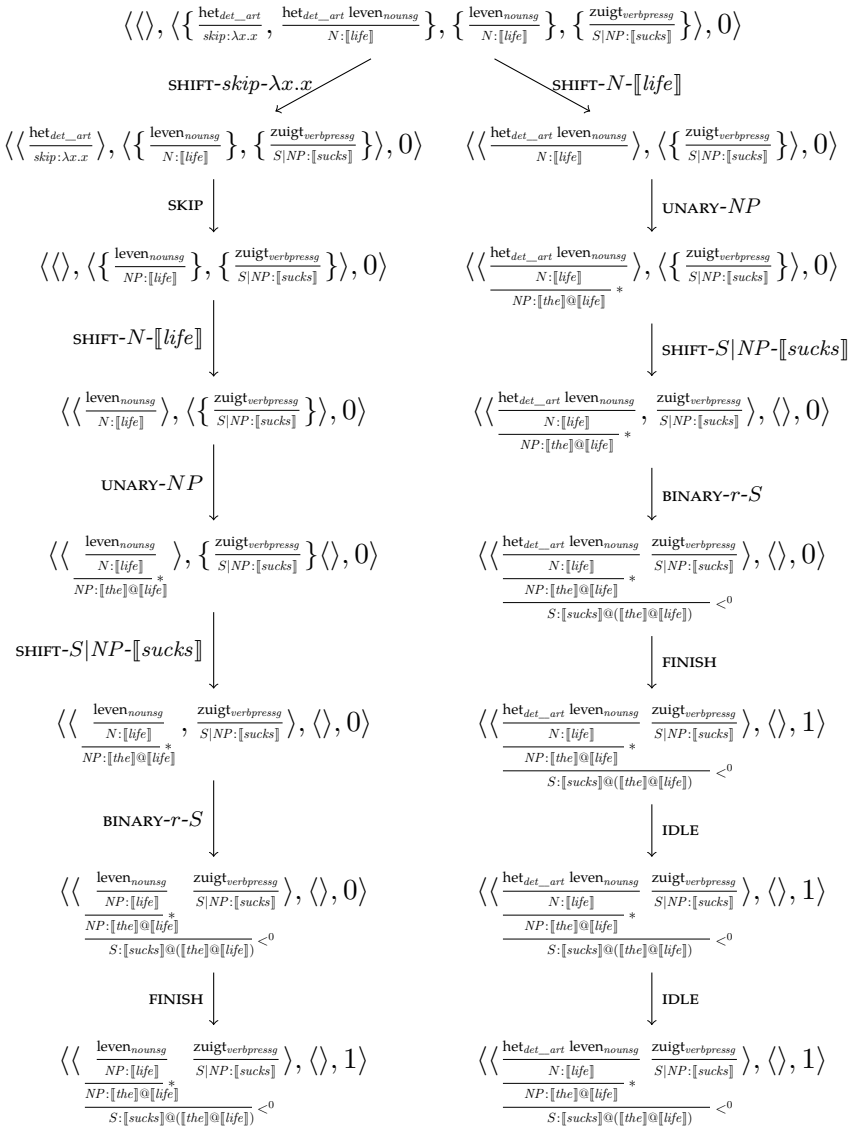


Figure 7.2: Parser items for the sentence *het leven zuigt* (*life sucks*).

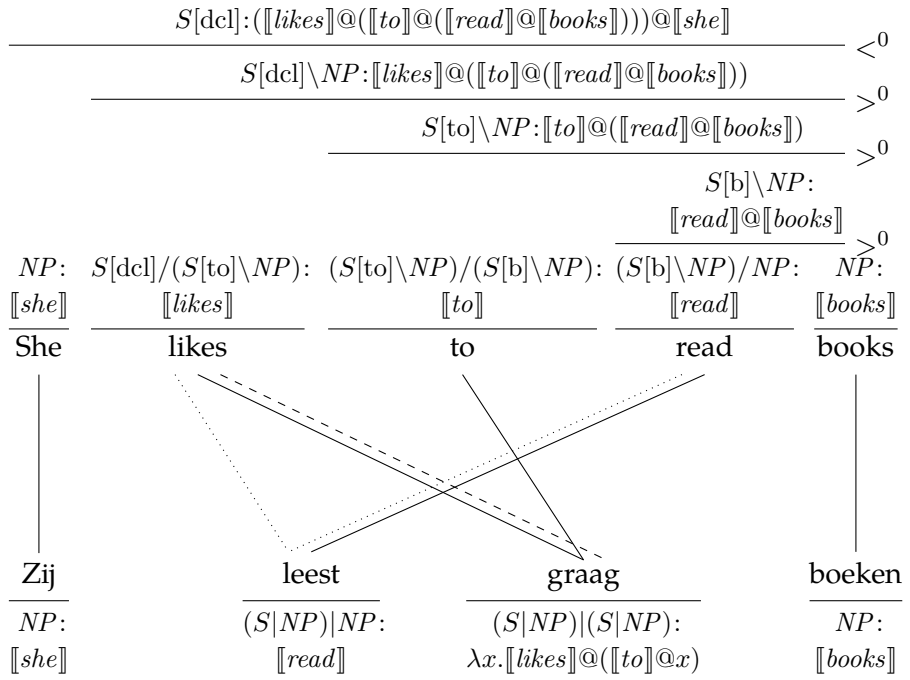


Figure 7.3: Category projection: word alignments induce candidate bilingual phrases, these induce candidate categories and interpretations for target-language words. Correct translation units are shown as solid lines, incorrect ones, dashed or dotted.

are said to belong to the same *generation*. Note that the parse on the right completes two generations earlier than the one on the left and uses the IDLE action twice to reach the last generation. Items not leading to a non-fragmentary parse are omitted for space reasons.

7.2.2 Step 1: Category Projection

Category projection assigns candidate categories and interpretations to target-language (multi)words in the training data. It thereby also induces the target-language lexicon that we use in the subsequent steps.

We first use a word alignment tool to create a set of *candidate bilingual phrases* $\mathcal{BP}$. Given a source-language sentence $e_1 e_2 \cdots e_m$ and its target-language translation $f_1 f_2 \cdots f_n$, a bilingual phrase is of the form $\langle [i,j],[k,l] \rangle$, $1 \leq i \leq j \leq m$, $1 \leq k \leq l \leq n$, i.e., it consists of an interval of source sentence word indices and an interval of target sentence word indices. It represents the hypothesis that words $e_i e_{i+1} \ldots e_j$ and words $f_k f_{k+1} \ldots f_l$ are phrases with the same meaning. Phrases can be derived from word alignments using sophisticated techniques (Och and Ney, 2004), but for now we use only phrases corresponding to (contiguous) translation units, thus $i = j \vee k = l$. Specifically, we use the N best alignments according to GIZA++ for both translation directions. An example of a sentence pair with translation units from multiple alternative word alignments is shown in Figure 7.3. For this example,

$$\mathcal{BP} \quad = \quad \{\langle [1],[1] \rangle, \langle [2,3],[3] \rangle, \langle [2],[3] \rangle, \langle [4],[2] \rangle, \langle [5],[4] \rangle\}$$

We then find the category and interpretation corresponding to each bilingual phrase. This is essentially the COMPOSE step in ARFCOISTRAP: we combine the words in $e_i e_{i+1} \ldots e_j$ using combinatory rules to obtain a single constituent with a category and an interpretation. If $i = j$, this is trivially the lexical constituent. Otherwise, we run the shift-reduce parser to find such a constituent. The combinatory rule instances used for parsing here are simply all that occur in any source-language derivation for the training data. We denote the first category and interpretation found for the phrase $e_i e_{i+1} \ldots e_j$, if any, as C_i^j and I_i^j, respectively.

For every candidate bilingual phrase $\langle [i,j],[k,l] \rangle \in \mathcal{BP}$ for which C_i^j and I_i^j are defined, $f_k f_{k+1} \ldots f_l$ is assigned a candidate lexical constituent with category $ConvCat(C_i^j)$ and interpretation I_i^j. The function $ConvCat$ turns the source-language categories into the categories we will use for parsing the target language. It replaces all slashes in a category with |, eliminating directionality because word order might be different in the target language. We also define it to remove all syntactic features such as $[\mathrm{dcl}]$ or $[\mathrm{pt}]$ (see Section 2.3.1) because these are specifically designed for English whereas featureless categories are thought to be more universal and to closely correspond to semantic types. We

found in initial experimentation that leaving the features out slightly improved results, so we adopted this practice. Target-language words f_k that are unaligned, i.e., not part of any bilingual phrase, are assigned the special category $skip$ and the identity interpretation $\lambda x.x$. In our example, five candidate lexical constituents are assigned (one for each candidate bilingual phrase), four of which are shown at the bottom of Figure 7.3.

The target-language lexicon L_f that we extract is, to a first approximation, the set of all lexical items $L := C : I$ induced by the training corpus in this manner. However, to prevent too many noisy items in the lexicon, we apply a cutoff with factor c. That is, we only include those items where the category/interpretation combination $C : I$ occurs at least c times as often with the (multiword) L as the most frequent category/interpretation combination for L—counting all N alignments for all training examples and both alignment directions.

7.2.3 Step 2: Derivation Projection

Derivation projection tries to build derivations for the target-language sentences that have the same interpretations as the English translations, using the candidate target-language lexical items found in category projection. For each training example, we only use the items extracted from that example.[1] Thus, the initial queue for the shift-reduce parser is defined as $Q = \langle q_1, q_2, \ldots, q_n \rangle$ where for $1 \leq k \leq n$,

$$
\begin{aligned}
q_k \quad = \quad & \{ \tfrac{f_k f_{k+1} \cdots f_l}{ConvCat(C_i^j):I_i^j} | \langle [i,j],[k,l] \rangle \in \mathcal{BP} \text{ and } C_i^j \text{ is defined} \} \cup \\
& \{ \tfrac{f_k}{skip:\lambda x.x} | \text{there is no } \langle [i,j],[k,l] \rangle \in \mathcal{BP} \text{ such that } C_i^j \text{ is defined} \}
\end{aligned}
$$

We must then find a derivation—or derivations—for the target-language sentence, based on Q and subject to the constraint that the interpretation of the derivation be the same as that of the given derivation for the source-language sentence.

[1] This includes some lexical items that did not make the cutoff and are not in L_f. It would make sense to exclude those, however, we found it does not improve results, so we left them in.

Input: A parse item $i = \langle\langle \overset{\cdots}{\overline{C_1{:}I_1}}, \overset{\cdots}{\overline{C_2{:}I_2}}, \ldots, \overset{\cdots}{\overline{C_k{:}I_k}}\rangle, Q, F\rangle$
Input: A desired interpretation I
Output: 1 if i can lead to a finished item with a single derivation with interpretation I on the stack, 0 otherwise
if $F = 1$ **then**
  if $k = 1 \wedge I_k \equiv I$ **then**
    return 1;
  end
  else
    return 0;
  end
end
else
  return $vsf(I_k, I)$;
end

Algorithm 7.1: ItemCheck: checking parse items against the desired sentence interpretation.

The search space is generally too large to enumerate all finished items. A first remedy is to prune away subtrees of the search tree that cannot contain an item with the desired interpretation. For this, we use the "valid semantic fragment" function vsf defined in Section 6.2.1. Our function for checking items additionally requires that finished items contain a single derivation whose interpretation is exactly equivalent to the desired sentence interpretation. It is given as Algorithm 7.1.

Despite the pruning, sometimes the search space is still too large, so a limit b' governs how large the agenda is allowed to get. If it grows larger, our algorithm aborts, returning no derivations.

The algorithm for finding target-language derivations is given as Algorithm 7.2. We call it ForceDecode because we decode (parse) while forcing the parser to find results with a certain interpretation. The idea has previously been employed in semantic parsing by Zhao and Huang (2015). Unlike them, we employ an agenda limit instead of a timeout in order to make results more strictly replicable.

Input: An initial queue Q_0
Input: A target interpretation I
Input: Sets of rule instances U, B
Input: A agenda limit b'
Output: The set of non-fragmentary derivations with interpretation I, or $\emptyset$ if the agenda limit is hit
$agenda \leftarrow \{\langle\langle\rangle, Q_0, 0\rangle\}$;
while *agenda contains unfinished items* **do**
  if $|agenda| > b'$ **then**
    return $\emptyset$;
  end
  else
    $agenda \leftarrow \{j | i \in agenda \wedge j \in succ_{U,B}(i) \wedge \textsc{ItemCheck}(j, I) = 1\}$;
  end
end
return $agenda$

Algorithm 7.2: ForceDecode

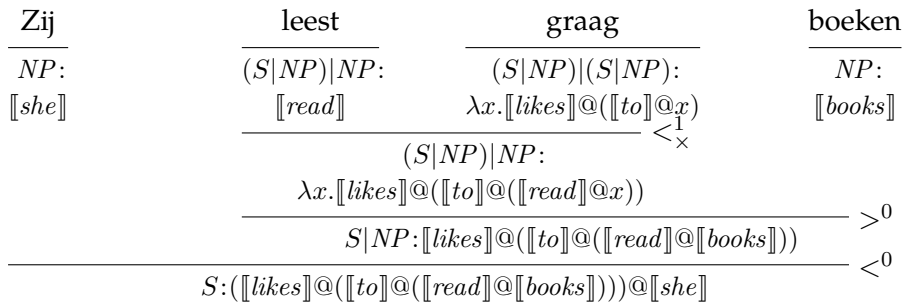


Figure 7.4: Derivation projection: combinatory rules are applied with the constraint of finding a derivation with the same interpretation as the source-language sentence.

The sets of rule instances U_f, B_f given to ForceDecode must be suitable for the target language and take into account that we verticalized all slashes and removed syntactic features. They are constructed from the source-language rule instance sets U_e, B_e as follows:

$$\begin{aligned} U_f &= \{ConvCat(Y){:}I \Rightarrow ConvCat(Z){:}K \mid Y{:}I \Rightarrow Z{:}K \in U_e\} \\ B_f &= \{ConvCat(X){:}I \;\; ConvCat(Y){:}J \Rightarrow ConvCat(Z){:}K, \\ &\quad\; ConvCat(Y){:}J \;\; ConvCat(X){:}I \Rightarrow ConvCat(Z){:}K \\ &\quad \mid X{:}I \;\; Y{:}J \Rightarrow Z{:}K \in B_e\} \end{aligned}$$

Note that *two* new rule instances in B_f correspond to each old rule instance in B_e: one with the original order of input categories, and one with the reversed one, implementing the intended meaning of the vertical slash—both orders are allowed. Note also that this does not quite double the number of rule instances because $ConvCat$ maps some categories to the same category. For example, the two rule instances in (1) both give rise to the same set of verticalized instances, namely that in (2).

(1) a. $(S/S){:}f \;\; S{:}a \Rightarrow S{:}(f@a)$
b. $S{:}a \;\; (S\backslash S){:}f \Rightarrow S{:}(f@a)$

(2) a. $(S|S){:}f \;\; S{:}a \Rightarrow S{:}(f@a)$
b. $S{:}a \;\; (S|S){:}f \Rightarrow S{:}(f@a)$

Figure 7.4 shows a resulting projected derivation for our example sentence.

7.2.4 Step 3: Parser Learning

In parser learning, we use the target-language derivations found in derivation projection to train a semantic parser for the target language.

Every training derivation D produced in derivation projection defines a unique simple path from the root of the search tree to the highest-in-the-tree item $\langle\langle D\rangle, \langle\rangle, 1\rangle$, i.e., it defines a sequence of parsing actions. For a given training example, we consider each action sequence leading to a derivation with the target interpretation as "correct", others as

"incorrect". Accordingly, parse items lying on the path of some correct sequence are "correct", others "incorrect". The task is now to train a parsing model that can be applied to an unseen target-language sentence and, among the many possible parses, pick out the one most likely to be correct. With our transition-based parsing system, this amounts to picking a simple path from the root of the search tree, at each step picking the parsing action most likely to be a correct one.

We follow Zhang and Clark (2011) in defining a global linear model factored by parsing steps. In this model, every parsing step is represented as a set of features, and the model scores each such set of features individually. Each "step" consists of a predecessor item i and a parsing action a which is applied to i to produce a successor item j. To map a step to a set of features, we first define a number of *slots*. A slot picks out a certain part of the structure of a step, if that part exists for the step. We first define the following *macroslots*:

- Q_0, Q_1, Q_2, Q_3: pick out the words at the first four positions left to process in the sentence.
- S_3, S_2, S_1, S_0: pick out the four rightmost constituents on the stack.
- S_3U, S_2U, S_1U, S_0U: in case of unary constituents, pick out their daughters.
- S_3H, S_2H, S_1H, S_0H: in case of binary constituents, pick out their head daughters. The head daughter is the one that is the functor, except if that is a modifier, in which case the argument is the head daughter.
- S_3R, S_2R, S_1R, S_0R: in case of binary constituents where the head daughter is on the left, pick out their nonhead daughters.
- S_3L, S_2L, S_1L, S_0L: in case of binary constituents where the head daughter is on the right, pick out their nonhead daughters.

For every macroslot T, the following *microslots* are defined, picking out substructures of the structures picked out by the macroslot:

- Tc: picks out the category of a constituent.
- Tw: picks out the word form of (the head word of) an element. A lexical constituent immediately contains its head word. The head word of a unary constituent is that of its daughter. The head word of a binary constituent is that of its head daughter. In the case of constituents on the stack, the head word may be a multiword.
- Tp: picks out the part-of-speech tag of (the head word of) an element. I the case of constituents on the stack, the head word may be a multiword, and its part of speech a string of parts of speech.
- A: picks out the parsing action taken.

Feature template names are concatenations of microslot names, leaving out repeated macroslot names. We use the same templates as Zhang and Clark (2011), shown in Figure 7.5.

For a given parsing step consisting of item i and action a, a feature template extracts nothing if some of its macroslots do not extract anything. Otherwise, it extracts a feature that is the concatenation of the template name and all the values extracted by its microslots.

In addition to these features, if a is a SHIFT action, we also extract a lexical feature that is the concatenation of the shifted lexical constituent's (multi)word, part of speech, category and interpretation. The purpose is to enable the model to learn to prefer certain lexical entries over others, even if they are only semantically different.

We define the *feature extraction function* ϕ such that $\phi(i, a)$ returns the feature set of the parsing step as a vector of 0's and 1's, where globally every unique feature is mapped to a position in the vector.[2]

The feature vector $\Phi(j)$ of a parsing item j is defined as follows: it is $\vec{0}$ for the initial item. For any other item j, generated from predecessor item i with action a, $\Phi(j) = \Phi(i) + \phi(i, a)$, i.e., the sum of the feature vectors of all steps leading up to it.

[2]Since our implementation uses a hash kernel (Bohnet, 2010) for this mapping, that position in the feature vector is not necessarily unique, although it is unique for most if not all features occurring in practice.

Input: A list Y of pairs of sentences and sets of "correct" action sequences
Input: A CCG $\langle L, U, B \rangle$
Input: A feature function Φ mapping parse items to feature vectors
Input: A beam width b
Input: A number of iterations T
Output: A weight vector $\vec{w}$
$\vec{w} \leftarrow \vec{0}$;
$\vec{sum} \leftarrow \vec{0}$;
$count \leftarrow 0$;
foreach $t \in [1, T]$ **do**
  foreach $\langle w_1 w_2 \dots w_n, S \rangle \in Y$ **do**
    $Q \leftarrow \langle q_1, q_2, \dots, q_n \rangle$ where $q_i = \{\frac{w_i w_{i+1} \dots w_j}{C:I} |$
    $w_i w_{i+1} \dots w_j := C{:}I \in L, i \geq j \geq n\}$;
    $agenda \leftarrow \{\langle\langle\rangle, Q, 0\rangle\}$;
    while *agenda contains unfinished items* **do**
      $agenda' \leftarrow \{j | i \in agenda \wedge j \in succ_{U,B}(i)\}$;
      For every item $i \in agenda'$, calculate its score $\vec{w} \cdot \Phi(i)$;
      Drop all but the b highest-scoring items and the highest-scoring finished item from $agenda'$;
      if $\{i \in agenda' | i \text{ is correct according to } S\} = \emptyset$ **then**
        break; `// Early update`
      end
      else
        $agenda \leftarrow agenda'$;
      end
    end
    $i^* = \leftarrow \arg\max_{i \in agenda} \Phi(i) \cdot \vec{w}$;
    $i' \leftarrow \arg\max_{i \in \{j \in agenda | j \text{ is correct according to } S\}} \vec{w} \cdot \Phi(i)$;
    $\vec{w} \leftarrow \vec{w} + \Phi(i') - \Phi(i^*)$; `// perceptron update`
    $\vec{sum} \leftarrow \vec{sum} + \vec{w}$;
    $count \leftarrow count + 1$
  end
end
return $\vec{sum}/count$; `// averaged perceptron`

Algorithm 7.3: BeamTrain

S_0wpA, S_0cA, S_0pcA, S_0wcA,
S_1wpA, S_1cA, S_1pcA, S_1wcA,
S_2pcA, S_2wcA,
S_3pcA, S_3wcA,

Q_0wpA, Q_1wpA, Q_2wpA, Q_3wpA,

S_0LpcA, S_0LwcA, S_0RpcA, S_0RwcA, S_0UpcA, S_0UwcA,
S_1LpcA, S_1LwcA, S_1RpcA, S_1RwcA, S_1UpcA, S_1UwcA,

S_0wcS_1wcA, S_0cS_1wA, S_0wS_1cA, S_0cS_1cA,
S_0wcQ_0wpA, S_0cQ_0wpA, S_0wcQ_0pA, S_0cQ_0pA,
S_1wcQ_0wpA, S_1cQ_0wpA, S_1wcQ_0pA, S_1cQ_0pA,

$S_0wcS_1cQ_0pA$, $S_0cS_1wcQ_0pA$, $S_0cS_1cQ_0wpA$, $S_0cS_1cQ_0pA$, $S_0pS_1pQ_0pA$,
$S_0wcQ_0pQ_1pA$, $S_0cQ_0wpQ_1pA$, $S_0cQ_0pQ_1wpA$, $S_0cQ_0pQ_1pA$, $S_0pQ_0pQ_1pA$,
$S_0wcS_1cS_2cA$, $S_0cS_1wcS_2cA$, $S_0cS_1cS_2wcA$, $S_0cS_1cS_2cA$, $S_0pS_1pS_2pA$,

$S_0cS_0HcS_0LcA$, $S_0cS_0HcS_0RcA$,
$S_1cS_1HcS_1RcA$,
$S_0cS_0RcQ_0pA$, $S_0cS_0RcQ_0wA$,
$S_0cS_0LcS_1cA$, $S_0cS_0LcS_1wA$,
$S_0cS_1cS_1RcA$, $S_0wS_1cS_1RcA$

Figure 7.5: The feature templates used by our parsing model, adapted from Zhang and Clark (2011).

Given a weight vector $\vec{w}$ with the same length as the feature vectors, j then has a *score* $\vec{w} \cdot \Phi(j)$. For a good parsing model, we need to find a $\vec{w}$ such that correct items score high and incorrect items score low. We do this, again following (Zhang and Clark, 2011), using using the averaged perceptron (Collins, 2002), implemented in Algorithm 7.3. The weight vector starts out as $\vec{0}$ and is then updated iteratively. In each iteration, the search tree for each training example is first explored with beam search, keeping the b items on the agenda that score highest according to the current $\vec{w}$.

The exploration ends when only finished items remain on the agenda. If the highest-scoring of them is indeed correct, the model made the cor-

rect predictions in this instance. If there is a correct item i' but also an even higher-scoring incorrect item i^*, our model is probably scoring the features of i' too low and those of i^* too high. A perceptron update then counteracts this by increasing the weights for the former and decreasing those for the latter, simply by adding $\Phi(i')$ and subtracting $\Phi(i^*)$. Since this update is only possible if a correct item exists, we perform an *early update* (Collins and Roark, 2004) just before the last correct item would drop off the agenda, even if there are still unfinished items. The final weight vector is obtained by taking the mean of its values after every training example in every iteration. The grammar used for training is $\langle L_f, U_f, B_f \rangle$.

Input: A sentence $w_1 w_2 \ldots w_n$
Input: A CCG $\langle L, U, B \rangle$
Input: A weight vector $\vec{w}$
Input: A beam width b
Output: An item containing a (possibly fragmentary) derivation
$Q \leftarrow \langle q_1, q_2, \ldots, q_n \rangle$ where $q_i = \{\frac{w_i w_{i+1} \ldots w_j}{C:I} |$
$w_i w_{i+1} \ldots w_j := C{:}I \in L, i \geq j \geq n\}$;
$agenda \leftarrow \{\langle\langle\rangle, Q, 0\rangle\}$;
while *agenda contains unfinished items* **do**
  $agenda' \leftarrow \{j | i \in agenda \wedge j \in succ_{U,B}(i)\}$;
  For every item $i \in agenda'$, calculate its score $\Phi(i) \cdot \vec{w}$;
  Drop all but the b highest-scoring items and the
  highest-scoring finished item from $agenda'$;
  $agenda \leftarrow agenda'$;
end
return $\arg\max_{i \in agenda} \Phi(i) \cdot \vec{w}$;

Algorithm 7.4: BeamDecode

Once $\vec{w}$ is trained, it can be applied to unseen data, attempting to find the highest-scoring item for a given sentence. We also use beam search for this, as shown in Algorithm 7.4.

Test data invariably contains a number of out-of-vocabulary (OOV) words that do not have entries in L_f. We need to ensure that parses can nevertheless be found for sentences containing them, and that their in-

$$\begin{aligned} \text{boek}_{nounsg} &:= N{:}[\![book]\!] \\ \text{getuige}_{nounsg} &:= N{:}f \end{aligned}$$

where

$$[\![book]\!] = \lambda x. \boxed{\begin{array}{|c|}\hline \\ \hline \text{book.n.01}(x) \\ \hline \end{array}}$$

$$f = \lambda x. \boxed{\begin{array}{|c|}\hline \\ \hline \text{\_\_UNKNOWN\_\_}(x) \\ \hline \end{array}}$$

Figure 7.6: A regular lexical entry for a seen word and a schematic lexical entry for an unseen word.

terpretations are mostly correct as far as possible. We do this by adding a *schematic* lexical entry for every OOV word w, based on entries for similar words. Concretely, we look at all single-word entries $w' := C{:}\, I' \in L_f$ where w' has the same part of speech as w. We add an auxiliary lexical entry $w := C{:}\, I$ for w for every such C, leaving it to the parsing model to choose the best category. To determine I, we look at all λ-DRSs I' and make them "schematic" by replacing names, synset IDs, roles, relations and cardinalities by the *__UNKNOWN__* symbol. The most frequent schematic interpretation thus generated is selected as I. For example, Figure 7.6 shows a lexical entry for the word *boek* ("book") whose λ-DRS shares a common structure with the majority of single-word entries with part-of-speech $nounsg$ and category N. The OOV word *getuige* ("witness") is accordingly assigned a DRS with the same structure and a dummy synset ID. The grammar used for decoding is thus $\langle L'_f, U_f, B_f \rangle$ where L'_f is L_f extended with the schematic lexical entries for unknown words in the test data.

7.3 Experiments and Results

7.3.1 Data and Source-language System

We have seen in Chapter 6 that our approach of projecting CCG derivations from one language to the other can handle a wide range of translation divergences. However, it does build on the assumption that the meanings of parallel sentences are exactly the same, so to generate high-quality training data with it, sentence pairs should be relatively literal translations of each other, as opposed to informative or loose translations (Bos, 2014) or translations using complex idiomatic expressions. Such literal pairs can be found, for instance, in resources aimed at human language learners, which pair example sentences with translations in a language that the learner is already familiar with. One such resource is Tatoeba (`https://tatoeba.org`), based on the Tanaka corpus (Tanaka, 2001).

We use 16,404 English-Dutch sentence pairs from Tatoeba, randomly divided into 13,122 for training, 1,639 for development and 1,641 as final test set, of which a random sample of 150 sentences was manually annotated to serve as a gold standard.

The source language system whose output we use for supervision is the C&C/Boxer system (Curran et al., 2007), which takes English sentences and produces CCG derivations with λ-DRS interpretations and which is also used for producing the Groningen Meaning Bank (cf. Section 5.4). We use the SVN repository version 2444, giving the options `--modal true --nn true --roles verbnet` to best match the annotation scheme described in Section 5.2. Additionally, we made some minor modifications to Boxer's code to better match the annotation scheme for adjectives, adverbs, semantic roles and modals. Using this system, we automatically annotated the English half of the data. The Dutch sentences were POS-tagged using TreeTagger (Schmid, 1994, 1995).

7.3.2 Evaluation Setup

Evaluation Metric How automatically produced meaning representations should be evaluated is an open question. However, a common

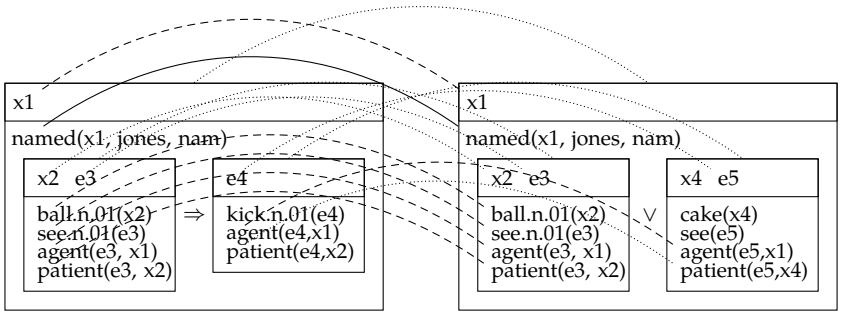


Figure 7.7: Two example DRSs being compared. The example is adapted from Le (2012). Matched boxes and referents are connected with dotted lines. Matched conditions are connected with dashed lines, or solid lines if they also appear under the same operator.

practice is to convert them into graphs and measure recall and precision based on overlap between the expected graph and the obtained graph. This was proposed by Allen et al. (2008) and implemented for Abstract Meaning Representations by Cai and Knight (2013) and for DRSs by Le and Zuidema (2012). We use the same metric and implementation as Le and Zuidema. The metric, which we dub the *DRS graph measure*, can be motivated as follows: we want to measure two things, firstly, that the right predications are made about the discourse referents, and secondly, that they are made in the right logical context, i.e., embedded under the right logical operator like negation, disjunction or implication. Intuitively, the DRS graph measure does this by assigning each DRS condition shared between expected DRS and obtained DRS one point, and one additional point if it appears under the same operator in both DRSs. This is illustrated in Figure 7.7.

The output of our parser is a stack with one complete derivation or several fragmentary derivations, each with a λ-DRS interpretation. We convert these interpretations into a graph T (in case of fragmentary derivations, the individual DRSs become connected components of a disconnected T). The expected DRS G is converted into a graph

G. Every (possibly nested) DRS, condition and referent is represented by a node. Condition nodes are labeled with the corresponding predicate or operator. Where a referent or DRS appears as an argument of a condition, this is indicated by an edge from the referent or box node to the condition node, labeled with the argument number (1 or 2). In which box a condition appears is indicated by an unlabeled edge from the condition node to the box node.

For graphs G_1 and G_2, $\Omega(G_1, G_2)$ is defined as the number of points that G_2 scores with respect to G_1. The function computes a maximum common subgraph isomorphism f from G_2 to G_1 where nodes can only be matched with nodes of the same type and nodes and edges with labels can only be matched with nodes/edges with the same label. Note that box nodes and referent nodes are unlabeled (they are labeled in the DRS, but these labels are arbitrary) and therefore have to be matched based on the structure of the graphs alone. We say that a condition node c in G_2 is *counted* if $f(c)$ is defined and for every argument a of c, $f(a)$ is the corresponding argument of $f(c)$ in G_1. We say that a condition node c in G_2 is *position-counted* if it is counted and has a box predecessor b such that $f(c)$ has a box predecessor $f(b)$ in G_1 that is counted. G_2 gets one point for each counted node in it, and one additional point for each position-counted node. $\Omega(G_1, G_2)$ is then the total number of points.

For a system output graph T and a target graph G, recall, precision and f-score are computed as follows:

$$
\begin{aligned}
\mathit{rec} &= \frac{\Omega(G,T)}{\Omega(G,G)} \\
\mathit{prec} &= \frac{\Omega(G,T)}{\Omega(T,T)} \\
\mathit{f1} &= \frac{2\Omega(G,T)}{\Omega(G,G)+\Omega(T,T)}
\end{aligned}
$$

For the example in Figure 7.7:

$$
\begin{aligned}
\textit{rec} &= \frac{\Omega(G,T)}{\Omega(G,G)} = \frac{8}{18} = 0.\overline{4} \\
\textit{prec} &= \frac{\Omega(G,T)}{\Omega(T,T)} = \frac{8}{20} = 0.4 \\
\textit{f1} &= \frac{2\Omega(G,T)}{\Omega(G,G)+\Omega(T,T)} = \frac{2 \cdot 8}{18+20} \approx 0.4211
\end{aligned}
$$

Baseline No comparable systems for Dutch as input language and DRT as meaning representation language exist yet. To demonstrate the effect of learning the parsing model, we defined a simple but informed baseline system to compare our system against. This baseline does not do any parsing but assigns target-language sentences interpretations by assigning each word its most frequent candidate interpretation, as determined in category projection (counting instances of candidate bilingual phrases from the N-best alignments used). The resulting semantic graph is highly fragmented.

Silver Standard We first measure how closely the output of our system for Dutch resembles the output of C&C/Boxer for English on the development/testing data. This gives an idea of how well our system has learned to imitate the existing system, but has two problems: first, it does not say much about the quality of the output because that of C&C/Boxer is not free from errors, it is not a gold standard. Secondly, the data contains idiomatic, informative and loose translations, in which case we *want* the output of both systems to differ.

Gold Standard Therefore, we also measure how closely the outputs of C&C/Boxer and our system resemble a hand-corrected gold standard of 150 sentence/DRS pairs from the testing portion, for their respective input languages. The gold standard was created as follows. Two annotators independently corrected 50 DRSs produced by C&C/Boxer so that the DRSs represented the meaning of Dutch annotations. Inter-annotator agreement at this point as measured by the evaluation met-

ric was 67% f-score. Instances of disagreement were identified, with 29% related to WordNet senses, 22% to semantic roles, 16% to other relations such as prepositional ones, 13% to the rendering of Dutch idioms using English WordNet senses, 9% to modal and logical operators such as implication and negation, and 11% to other structural issues such as nested DRSs. In an adjudication phase, both annotators resolved the differences together and agreed on a common gold standard. A single annotator then corrected another 100 Boxer DRSs, which were subsequently checked by the other annotator, and differences were again resolved through discussion. One annotator finally also created an adapted version of all 150 DRSs where in case of idiomatic, informative or loose translations, the annotation matches the English rather than Dutch sentence.

7.3.3 Experimental Setup

Intuitively, the more training examples we can use, the higher our system should perform in the end. How many training examples can be used depends on n, the number of n-best alignments we use for category projection, and the agenda limit b' for derivation projection. In initial trials, we experimented with different values for b'. The higher it is, the more sentence pairs receive at least one target-language derivation and can be used for training. However, we found that increasing the limit beyond 256 did not improve final results much and therefore used this value for all following experiments.

We then experimented with different values for n, the number of alignments used for category projection. Too few, and the lexical items necessary for successful derivation projection may not be generated for many training examples. Too many, and the parsing agenda is polluted with many incorrect lexical items, and derivation projection aborts due to the agenda limit b'. Figure 7.8 shows how the number of usable training examples varies as a function of n.

With the target-language derivations found, we train our parser for the target language on it, using Algorithm 7.3 (BeamTrain). We follow Zhang and Clark (2011) in using a beam width of $b = 16$ and initializing all model parameters to 0. We measure performance as f-score ac-

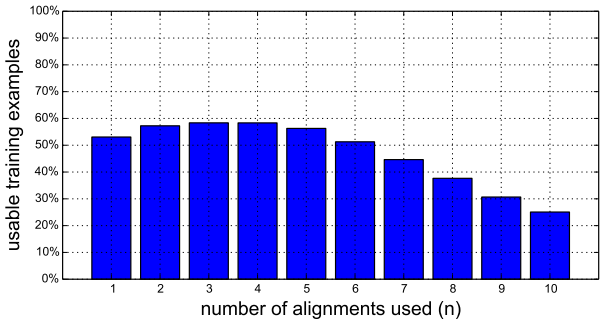


Figure 7.8: Percentage of successfully projected training examples depending on the number of n-best alignments (with $b' = 256$).

cording to the graph match measure, on the silver-standard annotated development data. The three hyperparameters influencing results are n, the lexical cutoff factor c and the number of training iterations T. Results in Figure 7.8 suggest that $n = 3$ is the optimal value for n, but more training examples may not necessarily translate to better performance. So we included all three hyperparameters in a parameter sweep on the development data, whose results are shown in Figure 7.9. We use the highest-scoring model found in the sweep ($n = 3$, $c = 0.2$, $T = 9$) for final testing on the 150-sentence gold standard.

7.3.4 Results and Discussion

Inspecting the results of the parameter sweep in Figure 7.9, it is clear that our semantic parser learns a lot in the first training iteration, after which learning is slow and not substantial after around the fifth iteration.

Table 7.1 shows the results of evaluating the highest-scoring found model on the gold-standard annotated test data, comparing the performance of our cross-lingually learned system on Dutch against the baseline and against C&C/Boxer's performance on the English versions of the same sentences.

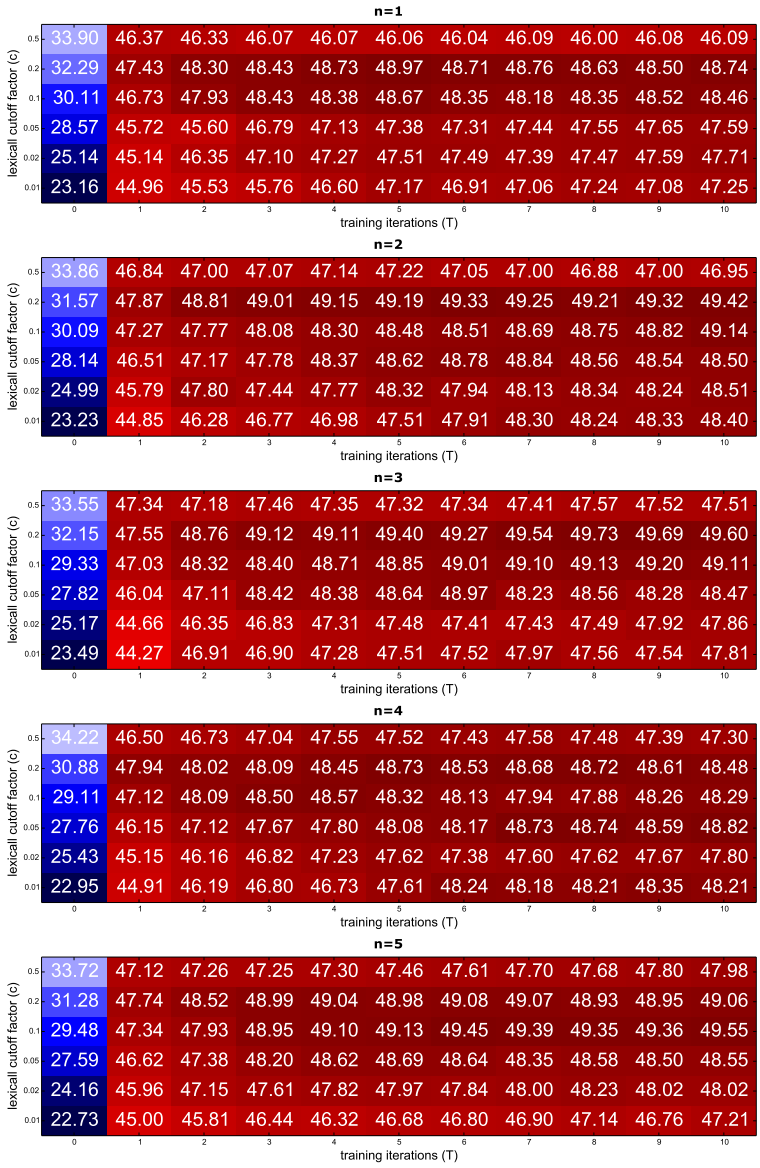


Figure 7.9: Development set f-score (in percent) depending on n-best alignments, lexical cutoff factor c and training iterations T.

Table 7.1: Gold-standard match f-score for Boxer, our baseline and our best cross-lingually trained model.

Language	English	Dutch	
System	C&C/Boxer	Baseline	Our system
Full	58.40	26.71	42.99
Ignoring WordNet senses	69.06	36.67	60.23
Ignoring VerbNet/LIRICS roles	64.51	27.57	47.82
Ignoring other relation labels	59.18	27.57	43.39
Ignoring all	78.88	39.04	69.22

C&C/Boxer obtains an f-score of 58.40% on the gold standard. Although the data, the meaning representation language and the evaluation metric are not directly comparable, we note that this f-score is in the same ballpark as current state-of-the-art open-domain semantic parsers for English, e.g., those that participated in the recent Abstract Meaning Representation Shared Task (May, 2016). A large part of the errors come from misidentifying word senses and semantic roles: "sloppy" evaluations in which we treat all word senses, all roles and/or other relation labels as equal give markedly higher f-scores of up to 78.88%.

Our learned system for Dutch scores around 15% lower than the source-language system under the strict evaluation, at 42.99%. The gap narrows to around 10% under the sloppy evaluation, where our system scores up to 69.22%. The gap is expected for a number of reasons. Firstly, the English system has the advantage of being based on an explicitly supervised syntactic parser. Secondly, the English system has access to the full WordNet lexicon while the Dutch system only has access to the words seen in the training data, resulting in many OOV items at test time. This helps explain the especially large gap under the strict evaluation. Thirdly, the system has to deal with noise from a number of sources, including non-literal translations in the training data, noisy word alignments, noisy lexical items resulting from the limitations of our projection method (discussed in Chapter 6) and automatically gen-

erated hence noisy training data. Let us discuss in turn how each of these challenges could be tackled in future work.

Lack of a Robust Syntactic Parser Given the great success of statistical syntactic parsing over the past decades and the very high accuracies obtained by state-of-the-art systems, it is not surprising that practically all broad-coverage semantic parsers to date rely on an external syntactic parser to constrain the search space or to provide features (cf. Section 3.7). These parsers are typically trained on datasets orders of magnitudes larger than the automatically syntactically and semantically annotated training dataset we used here. To a certain degree, doing without such a syntactic parser is unavoidable for the task we tackled here, because the aim was to develop a method applicable to under-resourced languages, implying that no such parser or data sufficient to train it exists. Nevertheless, if we want to apply our method to languages for which a robust syntactic parser *is* available[3], we could extend it by using externally provided syntactic features to guide derivation projection and the parser itself. Alternatively, one could train a system to assemble meaning representations directly from target-language syntactic parses, without ever generating target-language derivations for training.

OOV Items The problem of many unknown words at test time could be addressed by (partially) separating lexical learning from grammar learning (cf. Section 3.6). For example, one could mine larger amounts of parallel (or comparable) text for pairs of target-language words and WordNet senses. This could be done even with text that is too loosely translated or too syntactically complex to work well with our current derivation projection method.

Noisy Word Alignments The word alignments we use for category projection are not all correct. In using multiple alternative word alignments, we have designed category projection for recall, not precision. This may generate faulty lexical entries, but we count on derivation projection not being able to find derivations with the target interpretation

[3] As is in fact the case for Dutch, see for example Bouma et al. (2000).

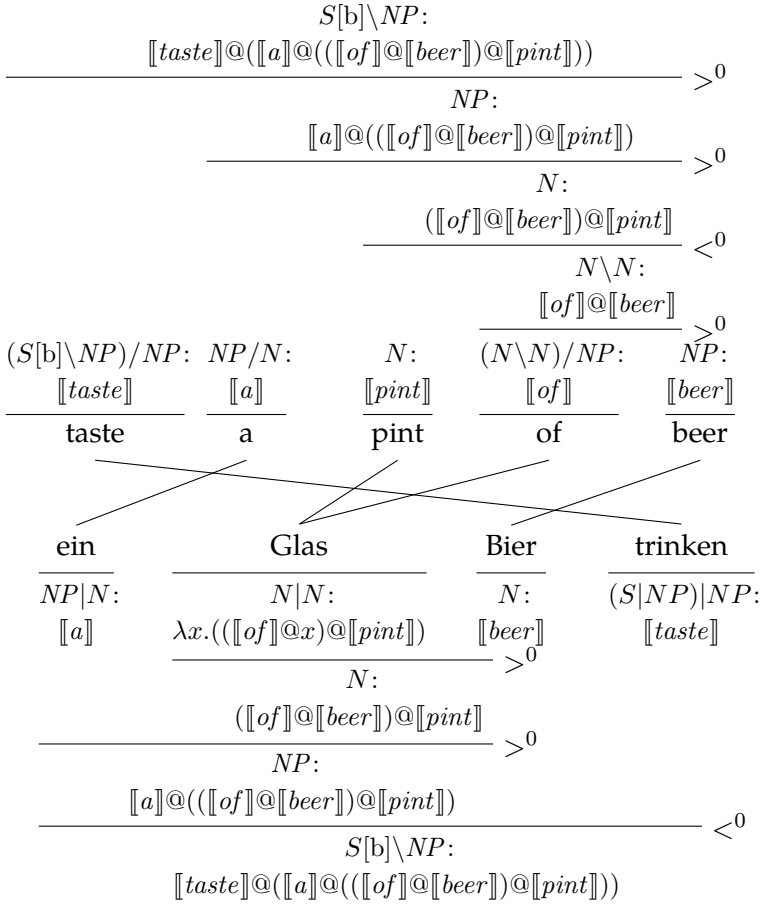


Figure 7.10: Derivation projection "succeeding" for a loose translation, producing incorrect lexical entries for German: *Glass* means *glass*, not *pint*, and *trinken* means *drink*, not *taste*. Example from Bos (2014).

using faulty lexical entries, or on parser learning assigning low weights to those entries. However, it can happen that derivation projection happily succeeds in building a target-language derivation with the "correct" interpretation, but using wrong lexical entries. Presumably, this is especially likely with structurally similar, but non-literal translations, e.g., with idioms like *kick the bucket / den Löffel abgeben* (cf. Section 6.1.1), or with informative or loose translations. An example of such an "infelicitous" parallel derivation is shown in Figure 7.10. Moreover, in part due to the noisy word alignments, if derivation projection succeeds, it typically produces more than one possible translation, resulting in multiple "correct" parser items in every generation. It is not clear in how far this ambiguity hampers parser learning. Evaluating category projection and derivation projection in-vitro could shed some light on possible ways to address such problems.

Limitations of Derivation Projection In Chapter 6, we identified a number of sub-types of translation divergences where our automatic approach to derivation projection produces analyses that do not correspond to the distinctions and generalizations that a grammar engineer would make. This results either in failing derivation projections and therefore a loss of training data, or in training derivations using lexical items that generalize poorly and therefore presumably hamper parser learning. Therefore, improving category projection and derivation projection may be an important area for improving the overall cross-lingual learning approach. For example, projected lexical entries could switch to a factored representation à la (Kwiatkowski et al., 2011; Wang et al., 2014), with categories and interpretation schemas stored separately from specific non-logical symbols. This would have the advantage of being able to project interpretations while possibly using different categories that may match the target-language construction better. It may also help cross-lingual semantic parsing in the same way it helps mono-lingual parsing, namely by enabling generalizations that make learning more efficient.

Automatically Generated Training Data Outside of category projection and derivation projection, which themselves are noisy, additional noise is introduced by the system that is used to create the source-language derivations. Higher-quality training data could be obtained by manually annotating the source-language side of parallel corpora. The more translations into different languages exist, the more semantic parsers could then be bootstrapped with our method. One recent effort to systematically annotate a multilingual parallel corpus with deep meaning representations is the Parallel Meaning Bank (PMB; Bjerva et al., 2014).

7.4 Conclusions

We have presented a method for training a broad-coverage semantic parser for a target language cross-lingually by projecting CCG derivations from a source language to target-language translations. The projection method proceeds in two steps, *category projection* and *derivation projection*, designed to handle the uncertainty resulting from automatically induced word alignments. The trained parser is a standard shift-reduce CCG parser with some extensions, e.g., to skip words and to handle multiwords.

We trained a system for Dutch which achieves an f-score of 42.99% under our strictest evaluation metric, compared to 58.40% for the corresponding English system. The large gap is unsurprising, as our method does not have access to a comparable supervised parser or comparable lexical resources as the source-language system, and in addition has to deal with multiple sources of noise. As we have outlined, there are a number of ways to address these challenges, including unsupervised lexical acquisition from large amounts of parallel data, investigation of error sources in category projection and derivation projection, factored lexicons as well as systematic manual annotation of data with multilingual translations.

Despite the challenges remaining to be addressed, the method is still potentially useful for target languages that lack adequate manually created computational grammars, lexical and annotated resources

and where the creation of such grammars and resources would be too expensive. It can also be used to bootstrap a semantic parser which is then improved through manual engineering and annotation.

Part IV

Conclusions

Chapter 8

Conclusions

Cross-lingual learning is the training of machine learning models for processing of a natural *target language* while using few or no manually created annotations or other resources for that target language—instead using such resources for a *source language* and using parallel corpora to project information from the source language to the target language. Cross-lingual learning is attractive because the creation of linguistic resources is very expensive at scale, and repeating this effort for multiple languages is often infeasible. In practice, resource creation efforts typically focus on a few, widely-spoken languages like English, leaving other languages without comparable natural-language processing tools. Cross-lingual learning can help bridge this gap.

The gap certainly exists for semantic parsing, i.e., learning to map natural-language utterances to formal meaning representations. Work to date has almost exclusively focused on English. In this thesis, we therefore considered the problem of learning a broad-coverage semantic parser cross-lingually.

We decided to use Combinatory Categorial Grammar (reviewed as background in Chapter 2) as a grammatical framework because it is designed to be applicable to all natural languages and maintains a high degree of transparency between syntactic categories and semantic types, which should make it helpful in abstracting away from cross-linguistic differences in syntactic constructions. CCG also has an ample body of

work in semantic parsing (reviewed in Chapter 3) on which we would be able to draw to develop our method.

This led to four research questions, which we are now ready to answer. Let us begin with the first one:

(i) Does CCG have the flexibility required for applying it to diverse natural languages, meaning representation formalisms and parsing strategies?

Seeing the broad range of semantic parsing tasks that CCG has successfully been applied to (cf. Chapter 3), we were already strongly inclined to answer this question in the affirmative.

In Chapter 4, we added another study to this body of work. It addressed a narrow-coverage task where natural-language instructions to a robot have to be mapped into a formal, special-purpose meaning representation language.

The task, although relatively small in scope, exhibits some of the typical challenges of semantic parsing, including dealing with unedited language, lexical ambiguity and a custom meaning representation language exhibiting different structural properties from natural language. We showed that CCG can be applied easily and successfully to induce a probabilistic grammar for the task, despite several aspects not covered in standard CCG, like custom basic categories, semantically empty words, a meaning representation language not based on the λ-calculus and interfacing with a spatial planner.

Our affirmative answer to the first research question thus holds up. Let us turn to the second one:

(ii) Broad-coverage semantic parsing requires training data in the form of text annotated with suitable meaning representations such as Discourse Representations Structures (DRS). How can the knowledge of humans be used effectively for building such a corpus?

We provided one answer to this question in Chapter 5 by describing how this is done in the Groningen Meaning Bank project: by combining automatic and human annotation decisions in a clever way. We

showed how this is facilitated by a mode of annotation where annotation decisions (called Bits of Wisdom or BOWs) from multiple sources are integrated dynamically into a natural-language processing toolchain, and where such decisions are largely at the token level. In particular, we showed that a purely lexical mode of annotating quantifier scope is powerful enough to obtain the desired reading in the vast majority of cases.

We found that it remains difficult to obtain a large number of annotations that can be considered gold standard, compared to other efforts which rely less on automatic or crowdsourced annotation and more on expert annotation, albeit in some cases targeting less "deep" meaning representations. The difficulty of annotation highlights the need for semantic parsers that require little explicit supervision to train, especially when the process is to be repeated for multiple natural languages. Our third research question addresses this:

(iii) One type of cross-lingual learning is *annotation projection*, the projection of source-language annotations to target-language annotations, followed by training on the target-language data so annotated. In the case of CCG derivations, annotation projection amounts to automatic parallel semantic treebanking:

- Is there an algorithm for doing this?
- Can it deal with translation divergences?
- Does it produce linguistically adequate analyses?

We proposed such an algorithm in Chapter 6, developed in several steps, looking at several ways in which target-language sentences differ from their source-language translations. We showed that it can successfully project many derivations. Evaluating it against the taxonomy of translation divergences presented by Dorr (1993), we found that it can deal with thematic, structural, categorial, head-switching and conflational divergences, albeit with certain limitations for each type. Some of the limitations cause projection to fail for certain instances, others lead to linguistically inadequate lexical items that are liable to generalize

poorly. Some of these shortcomings could be addressed, e.g., by learning to split more synthetic source-language constructions into more analytic target-language constructions, but given the variety of natural language, it is unlikely that such an algorithm would ever be completely error-free.

Our answer to the research question is thus that a derivation projection algorithm for CCG does exist, but cannot be relied upon to provide linguistically adequate parallel analyses in all cases. It can, however, be useful for providing a starting point for manual parallel treebanking, or for producing target-language training data, provided that the learning method is sufficiently robust to noise. This leads to our fourth and final research question:

(iv) How can such projected derivations be used to train a broad-coverage semantic parser?

We addressed this question in Chapter 7 by developing a method to train such a parser. The method is based on the projection algorithm developed in Chapter 6, but is modified in order to be able to use automatic, hence uncertain and noisy, word alignments. In two steps called *category projection* and *derivation projection*, a lexicon and training data for the target-language are generated, and a third step called *parser learning* trains a shift-reduce parser with a global linear model, using perceptron updates. Out-of-vocabulary words are dealt with at test time based on lexical semantic schemas.

We showed that the approach can train a semantic parser for the target language significantly outperforming a simple baseline. We also found, unsurprisingly, that the cross-lingually trained parser's performance still lags considerably behind parsers which are able to draw on gold-standard training data and lexical resources for their respective target language. However, we also pointed to a number of possible ways of improving our method while keeping its desirable property of relying on few target-language resources.

To conclude, we can thus say that cross-lingual training of broad-coverage semantic parsers does indeed work—to a degree. The accuracy of the cross-lingually trained Dutch parser presented herein is still

far from a point where one would want to use it for anything mission-critical. However, this is still true for broad-coverage semantic parsing in general, which remains a very challenging task even in the presence of manually created training data for the target language.

The methods for derivation projection and cross-lingual semantic parser training presented in this thesis can benefit future research in two main ways. First, as monolingual methods for learning semantic parsers improve, cross-lingual methods can help transfer these results to many languages more cheaply and quickly than it would be possible without them. Of course, *where possible*, manual semantic annotation for each target language is to be preferred because it will generally produce higher-quality training data than automatic projection. Thus, secondly, derivation projection and cross-lingually trained parsers can be used to assist with manual annotation, for example by providing initial automatic analyses and having annotators correct them.

Bibliography

Abend, Omri and Rappoport, Ari (2013). Universal conceptual cognitive annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238.

Allen, James (1994). *Natural Language Understanding*. Benjamin Cummings, 2nd edition.

Allen, James F., Swift, Mary, and de Beaumont, Will (2008). Deep Semantic Analysis of Text. In Bos, Johan and Delmonte, Rodolfo, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 343–354. College Publications.

Ambati, Bharat Ram, Deoskar, Tejaswini, Johnson, Mark, and Steedman, Mark (2015). An incremental algorithm for transition-based CCG parsing. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 53–63.

Andreas, Jacob, Vlachos, Andreas, and Clark, Stephen (2013). Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 47–52.

Andrew, Galen and MacCartney, Bill (2004). Statistical resolution of scope ambiguity in natural language. Unpublished manuscript.

Artzi, Yoav, Lee, Kenton, and Zettlemoyer, Luke (2015). Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710.

Artzi, Yoav and Zettlemoyer, Luke (2011). Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 421–432.

Artzi, Yoav and Zettlemoyer, Luke (2013a). UW SPF: The University of Washington Semantic Parsing Framework.

Artzi, Yoav and Zettlemoyer, Luke S. (2013b). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

Asher, Nicholas and Lascarides, Alex (2003). *Logics of conversation*. Studies in natural language processing. Cambridge University Press.

Auli, Michael and Lopez, Adam (2011). Training a log-linear parser with loss functions via softmax-margin. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 333–343.

Baldridge, Jason (2002). *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh.

Baldridge, Jason and Kruijff, Geert-Jan M. (2003). Multi-modal combinatory categorial grammar. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 211–218.

Banarescu, Laura, Bonial, Claire, Cai, Shu, Georgescu, Madalina, Griffitt, Kira, Hermjakob, Ulf, Knight, Kevin, Koehn, Philipp, Palmer, Martha, and Schneider, Nathan (2013). Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Barendregt, Henk (1984). *The Lambda calculus: its syntax and semantics*. North-Holland.

Basile, Valerio and Bos, Johan (2013). Aligning formal meaning representations with surface strings for wide-coverage text generation. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 1–9, Sofia, Bulgaria. Association for Computational Linguistics.

Basile, Valerio, Bos, Johan, Evang, Kilian, and Venhuizen, Noortje (2012a). A platform for collaborative semantic annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 92–96.

Basile, Valerio, Bos, Johan, Evang, Kilian, and Venhuizen, Noortje Joost (2012b). Developing a large semantically annotated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*.

Bates, Madeleine, Boisen, Sean, and Makhoul, John (1990). Developing an evaluation methodology for spoken language systems. In *Proceedings of the Third DARPA Speech and Natural Language Workshop*, pages 102–105.

Bech, Gunnar (1952). Über das niederländische Adverbialpronomen 'er'. *Travaux du cercle linguistique de Copenhague*, 8:5–32.

Bender, Emily M., Flickinger, Dan, Oepen, Stephan, Packard, Woodley, and Copestake, Ann (2015). Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 239–249.

Bennis, Hans (1986). *Gaps and Dummies*. Foris Publications, Dordrecht.

Berant, Jonathan, Chou, Andrew, Frostig, Roy, and Liang, Percy (2013). Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Berant, Jonathan and Liang, Percy (2014). Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1415–1425.

Berant, Jonathan and Liang, Percy (2015). Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Berends, Sanne, Veenstra, Alma, and van Hout, Angeliek (2010). 'Nee, ze heeft er twee': acquisition of the Dutch quantitative 'er'. *Groninger Arbeiten zur Germanistischen Linguistik*, 51:1–7.

Beschke, Sebastian, Liu, Yang, and Menzel, Wolfgang (2014). Large-scale CCG induction from the Groningen Meaning Bank. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 12–16.

Bird, Steven, Loper, Edward, and Klein, Ewan (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

Bjerva, Johannes (2014). Multi-class animacy classification with semantic features. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–75.

Bjerva, Johannes, Bos, Johan, and Haagsma, Hessel (2016). The Meaning Factory at SemEval-2016 task 8: Producing AMRs with Boxer. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1179–1184.

Bjerva, Johannes, Evang, Kilian, and Bos, Johan (2014). Towards a parallel meaning bank. Poster presented at the 24th Meeting of Computational Linguistics in the Netherlands.

Blackburn, Patrick and Bos, Johan (2005). *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI Publications.

Blackburn, Patrick, Bos, Johan, Kohlhase, Michael, and de Nivelle, Hans (1999). Inference and Computational Semantics. In Bunt, H.C. and Thijsse, E.G.C., editors, *Third International Workshop on Computational Semantics (IWCS-3)*, pages 5–19.

Bohnet, Bernd (2010). Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97.

Bohnet, Bernd and Nivre, Joakim (2012). A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465.

Bollacker, Kurt, Evans, Colin, Paritosh, Praveen, Sturge, Tim, and Taylor, Jamie (2008). Freebase: a collaboratively edited graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*.

Bonial, Claire, Corvey, William J., Palmer, Martha, Petukhova, Volha, and Bunt, Harry (2011). A hierarchical unification of LIRICS and VerbNet semantic roles. In *Proceedings of the 5th IEEE International Conference on Semantic Computing (ICSC 2011)*, pages 483–489.

Bos, Johan (1996). Predicate Logic Unplugged. In Dekker, P. and Stokhof, M., editors, *Proceedings of the Tenth Amsterdam Colloquium*, pages 133–143.

Bos, Johan (2004). Computational Semantics in Discourse: Underspecification, Resolution, and Inference. *Journal of Logic, Language and Information*, 13(2):139–157.

Bos, Johan (2008). Wide-Coverage Semantic Analysis with Boxer. In Bos, J. and Delmonte, R., editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 277–286. College Publications.

Bos, Johan (2009). Towards a large-scale formal semantic lexicon for text processing. In Chiarcos, Christian, Eckart de Castilho, Richard, and Stede, Manfred, editors, *Proceedings of the Biennal GSCL Conference 2009*, pages 3–14. Gunter Narr Verlag.

Bos, Johan (2014). Semantic annotation issues in parallel meaning banking. In Bunt, Harry, editor, *Proceedings 10th Joint ISO - ACL SIGSEM Workshop on Interoperable Semantic Annotation (ISA-10)*, pages 17–20.

Bos, Johan (2016). Expressive power of abstract meaning representations. *Computational Linguistics*, 42(3).

Bos, Johan, Basile, Valerio, Evang, Kilian, Venhuizen, Noortje, and Bjerva, Johannes (2017). The Groningen Meaning Bank. In Ide, Nancy and Pustejovsky, James, editors, *The Handbook of Linguistic Annotation*. Springer, Berlin. To appear.

Bos, Johan, Clark, Stephen, Steedman, Mark, Curran, James R., and Hockenmaier, Julia (2004). Wide-coverage semantic representations from a CCG parser. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1240–1246.

Bos, Johan, Evang, Kilian, and Nissim, Malvina (2012). Annotating semantic roles in a lexicalised grammar environment. In *Proceedings of isa-8, 8th Joint ISO - ACL SIGSEM Workshop on Interoperable Semantic Annotation*.

Bos, Johan and Markert, Katja (2005). Recognising textual entailment with logical inference. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 628–635.

Bos, Johan and Markert, Katja (2006). Recognising textual entailment with robust logical inference. In Quiñonero-Candela, Joaquin, Dagan, Ido, Magnini, Bernardo, and d'Alché Buc, Florence, editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment: First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, pages 404–426. Springer, Berlin, Heidelberg.

Bouma, Gosse, van Noord, Gertjan, and Malouf, Robert (2000). Alpino: Wide coverage computational analysis of dutch. In *Computational Linguistics in the Netherlands*.

Branavan, S.R.K., Zettlemoyer, Luke S., and Barzilay, Regina (2010). Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1268–1277.

Bresnan, Joan (2001). *Lexical-Functional Syntax*. Blackwell Publishing.

Brown, Peter F., Della Pietra, Vincent J., Della Pietra, Stephen A., and Mercer, Robert L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–313.

Butler, Alastair (2015). *Linguistic Expressions and Semantic Processing: A Practical Approach*. Springer, Heidelberg.

Butt, Miriam, Dyvik, Helge, King, Tracy Holloway, Masuichi, Hiroshi, and Rohrer, Christian (2002). The parallel grammar project. In *Proceedings of the COLING-2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7.

Börschinger, Benjamin, Jones, Bevan K., and Johnson, Mark (2011). Reducing grounded learning tasks to grammatical inference. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1416–1425.

Cai, Qingqing and Yates, Alexander (2013). Semantic parsing Freebase: Towards open-domain semantic parsing. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 328–338.

Cai, Shu and Knight, Kevin (2013). Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria.

Carnap, Rudolf (1947). *Meaning and Necessity: a Study in Semantics and Modal Logic*. The University of Chicago Press, Chicago.

Central Intelligence Agency (2006). *The CIA World Factbook*. Potomac Books.

Chen, David L. (2012). Fast online lexicon learning for grounded language acquisition. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 430–439.

Chen, David L. and Mooney, Raymond J. (2008). Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning*, pages 128–135.

Chen, David L. and Mooney, Raymond J. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, pages 859–865.

Chomsky, Noam (1988). *Lectures on Government and Binding: The Pisa Lectures*. Foris Publications.

Chomsky, Noam (1995). *The Minimalist Program*. MIT Press.

Church, Alonzo (1932). A set of postulates for the foundation of logic. *Annals of Mathematics*, 33(2):346–366.

Clark, Stephen and Curran, James R. (2004). Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Clark, Stephen and Curran, James R. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Clark, Stephen, Hockenmaier, Julia, and Steedman, Mark (2002). Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 327–334.

Clarke, James, Goldwasser, Dan, Chang, Ming-Wei, and Roth, Dan (2010). Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 18–27.

Collins, Michael (2002). Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, Philadelphia, USA.

Collins, Michael and Roark, Brian (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

Copestake, Ann and Flickinger, Dan (2000). An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*.

Copestake, Ann, Flickinger, Dan, Sag, Ivan, and Pollard, Carl (2005). Minimal recursion semantics: An introduction. *Journal of Research on Language and Computation*, 3(2–3):281–332.

Curran, James, Clark, Stephen, and Bos, Johan (2007). Linguistically Motivated Large-Scale NLP with C&C and Boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Companion Volume: Proceedings of the Demo and Poster Sessions*, pages 33–36.

Curran, James R. and Clark, Stephen (2003a). Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 11th Meeting of the European Chapter of the Association for Computational Linguistics*, pages 91–98.

Curran, James R. and Clark, Stephen (2003b). Language independent NER using a maximum entropy tagger. In *Proceedings of the Seventh conference on Natural Language Learning at HLT-NAACL 2003*, pages 164–167.

Dalrymple, Mary (2001). *Lexical Functional Grammar*, volume 34 of *Syntax and Semantics*. Academic Press.

Doran, Christine and Srinivas, Bangalore (2000). Developing a wide-coverage CCG system. In Abeillé, Anne and Rambow, Oren, editors, *Tree Adjoining Grammars. Formalisms, Linguistic Analysis and Processing*. CSLI Publications.

Dorr, Bonnie Jean (1993). *Machine Translation: A View from the Lexicon*. The MIT Press, Cambridge, Massachusetts.

Dryer, Matthew S. (2013a). Determining dominant word order. In Dryer, Matthew S. and Haspelmath, Martin, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Dryer, Matthew S. (2013b). Expression of pronominal subjects. In Dryer, Matthew S. and Haspelmath, Martin, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Dryer, Matthew S. (2013c). Order of adjective and noun. In Dryer, Matthew S. and Haspelmath, Martin, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Dryer, Matthew S. (2013d). Order of demonstrative and noun. In Dryer, Matthew S. and Haspelmath, Martin, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Dryer, Matthew S. (2013e). Order of relative clause and noun. In Dryer, Matthew S. and Haspelmath, Martin, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Dryer, Matthew S. (2013f). Order of subject, object and verb. In Dryer, Matthew S. and Haspelmath, Martin, editors, *The World Atlas of Lan-*

guage Structures Online. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Dukes, Kais (2013a). Semantic annotation of robotic spatial commands. In *Language and Technology Conference (LTC)*.

Dukes, Kais (2013b). Train robots: A dataset for natural language human-robot spatial interaction through verbal commands. In *International Conference on Social Robotics (ICSR). Embodied Communication of Goals and Intentions Workshop*.

Dukes, Kais (2014). SemEval-2014 task 6: Supervised semantic parsing of robotic spatial commands. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 45–53.

Egg, Markus, Koller, Alexander, and Niehren, Joachim (2001). The constraint language for lambda structures. *Logic, Language and Information*, 10:457–485.

Eisner, Jason (1996). Efficient normal-form parsing for combinatory categorial grammar. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 79–86.

Erman, Lee, editor (1977). *SIGART Newsletter*, volume 61. ACM, New York, NY, USA.

Evang, Kilian, Basile, Valerio, Chrupała, Grzegorz, and Bos, Johan (2013). Elephant: Sequence labeling for word and sentence segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1426.

Evang, Kilian and Bos, Johan (2013). Scope disambiguation as a tagging task. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Short Papers*, pages 314–320.

Evang, Kilian and Bos, Johan (2014). RoBox: CCG with structured perceptron for supervised semantic parsing of robotic spatial commands. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 482–486.

Evang, Kilian and Bos, Johan (2016). Cross-lingual learning of an open-domain semantic parser. In *Proceedings of COLING 2016, the 25th International Conference on Computational Linguistics*. To appear.

Fellbaum, Christiane, editor (1998). *WordNet. An Electronic Lexical Database*. The MIT Press.

Flanigan, Jeffrey, Thomson, Sam, Carbonell, Jaime, Dyer, Chris, and Smith, A. Noah (2014). A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436.

Flickinger, Dan, Kordoni, Valia, and Zhang, Yi (2012a). DeepBank: A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories (TLT11)*.

Flickinger, Dan, Kordoni, Valia, Zhang, Yi, Branco, António, Simov, Kiril, Osenova, Petya, Carvalheiro, Catarina, Costa, Francisco, and Castro, Sérgio (2012b). ParDeepBank: Multiple parallel deep treebanking. In *Proceedings of the Twelfth Workshop on Treebanks and Linguistic Theories (TLT12)*, pages 97–108.

Fowler, Timothy A. D. and Penn, Gerald (2010). Accurate context-free parsing with combinatory categorial grammar. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 335–344.

Ganchev, Kuzman, Gillenwater, Jennifer, and Taskar, Ben (2009). Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the OPTAFNLP: Volume 1*, pages 369–377.

Ge, Ruifang and Mooney, Raymond J. (2005). A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the 9th Conference on Computational Natural Language Learning*, pages 9–16.

Ge, Ruifang and Mooney, Raymond J. (2006). Discriminative reranking for semantic parsing. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 263–270.

Ge, Ruifang and Mooney, Raymond J. (2009). Learning a compositional semantic parser using an existing syntactic parser. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 611–619.

Goldwasser, Dan, Reichart, Roi, Clarke, James, and Roth, Dan (2011). Confidence driven unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 1486–1495.

Goldwasser, Dan and Roth, Dan (2011). Learning from natural instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Hautli, Annette and King, Tracy Holloway (2009). Adapting stochastic LFG input for semantics. In Butt, Miriam and King, Tracy Holloway, editors, *Proceedings of the LFG09 Conference*, pages 358–377. CSLI Publications.

He, Yulan and Young, Steve (2003). Hidden vector state model for hierarchical semantic parsing. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 268–271.

He, Yulan and Young, Steve (2005). Semantic processing using the Hidden Vector State model. *Computer Speech and Language*, 19:85–106.

He, Yulan and Young, Steve (2006). Spoken language understanding using the Hidden Vector State Model. *Speech Communication*, 48(3–4):262–275.

Higgins, Derrick and Sadock, Jerrold M. (2003). A machine learning approach to modeling scope preferences. *Computational Linguistics*, 29(1):73–96.

Hockenmaier, Julia (2003a). *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. PhD thesis, Univ. of Edinburgh.

Hockenmaier, Julia (2003b). Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 359–366.

Hockenmaier, Julia (2006). Creating a CCGbank and a wide-coverage CCG lexicon for german. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 505–512.

Hockenmaier, Julia, Bierner, Gann, and Baldridge, Jason (2000). Providing robustness for a CCG system. In *Proceedings of ESSLLI'2000 Workshop on Linguistic Theory and Grammar Implementation*.

Hockenmaier, Julia and Bisk, Yonatan (2010). Normal-form parsing for combinatory categorial grammars with generalized composition and type-raising. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 465–473.

Hockenmaier, J. and Steedman, M. (2002). Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 335–342.

Hockenmaier, J. and Steedman, M. (2007). CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Hoffman, Beryl (1995). *The Computational Analysis of the Syntax and Interpretation of "Free" Word Order in Turkish*. PhD thesis, University of Pennsylvania.

Honnibal, Matthew (2010). *Hat Categories: Representing Form and Function Simultaneously in Combinatory Categorial Grammar*. PhD thesis, University of Sydney.

Horn, Lawrence R. (1996). Exclusive company: *Only* and the dynamics of vertical inference. *Journal of Semantics*, 13:1–40.

Huet, G. P. (1975). A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1(1):27–57.

Huybregts, Riny (1976). Overlapping dependencies in Dutch. *Utrecht Working Papers in Linguistics*, 1:24–65.

Hwa, Rebecca, Resnik, Philip, Weinberg, Amy, Cabezas, Clara, and Kolak, Okan (2005). Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325.

Ide, Nancy, Fellbaum, Christiane, Baker, Collin, and Passonneau, Rebecca (2010). The manually annotated sub-corpus: a community resource for and by the people. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 68–73.

Janssen, Theo M.V. (2012). Compositionality: its historic context. In Werning, Markus, Hinzen, Wolfram, and Machery, Edouard, editors, *The Oxford Handbook of Compositionality*, pages 19–46. Oxford University Press, Oxford.

Jones, Bevan Keeley, Johnson, Mark, and Goldwater, Sharon (2012). Semantic parsing with Bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 488–496.

Kamp, Hans (1984). A Theory of Truth and Semantic Representation. In Groenendijk, Jeroen, Janssen, Theo M.V., and Stokhof, Martin, editors, *Truth, Interpretation and Information*, pages 1–41. FORIS, Dordrecht, Holland/Cinnaminson, U.S.A.

Kamp, Hans and Reyle, Uwe (1993). *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht.

Kate, Rohit (2014). UWM: Applying an existing trainable semantic parser to parse robotic spatial commands. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 823–827.

Kate, Rohit J. and Mooney, Raymond J. (2006). Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 913–920.

Kate, Rohit J. and Mooney, Raymond J. (2007). Learning language semantics from ambiguous supervision. In *Proceedings of the Twenty-second AAAI Conference on Artificial Intelligence*, pages 895–900.

Kate, Rohit J., Wong, Yuk Wah, and Mooney, Raymond J. (2005). Learning to transform natural to formal languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 1062–1068.

Keller, Frank (2010). Cognitively plausible models of human language processing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics: Short Papers*, pages 60–67.

Kim, Joohyun and Mooney, Raymond J. (2010). Generative alignment and semantic parsing for learning from ambiguous supervision. In *Coling 2010: Posters*, pages 543–551.

Kim, Joohyun and Mooney, Raymond J. (2012). Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 433–444.

Kipper, Karin, Korhonen, Anna, Ryant, Neville, and Palmer, Martha (2008). A large-scale classification of English verbs. *Language Resources and Evaluation*, 42(1):21–40.

Koehn, Philipp (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Koehn, Philipp, Och, Franz Josef, and Marcu, Daniel (2003). Statistical phrase-based translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics*, pages 48–54.

Krishnamurthy, Jayant and Mitchell, M. Tom (2015). Learning a compositional semantics for Freebase with an open predicate vocabulary. *Transactions of the Association of Computational Linguistics*, 3:257–270.

Krishnamurthy, Jayant and Mitchell, Tom (2012). Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765.

Kuhlmann, Marco, Koller, Alexander, and Satta, Giorgio (2015). Lexicalization and generative power in CCG. *Computational Linguistics*, 41(2):187–219.

Kuhn, Roland and de Mori, Renato (1995). The application of semantic classification trees to natural language understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):449–160.

Kwiatkowksi, Tom, Zettlemoyer, Luke, Goldwater, Sharon, and Steedman, Mark (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233.

Kwiatkowski, Tom, Choi, Eunsol, Artzi, Yoav, and Zettlemoyer, Luke (2013). Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556. Association for Computational Linguistics.

Kwiatkowski, Tom, Zettlemoyer, Luke, Goldwater, Sharon, and Steedman, Mark (2011). Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523.

Larson, Richard (2002). The grammar of intensionality. In Peter, Georg and Preyer, Gerhard, editors, *Logical From and Language*, pages 229–262. Oxford University Press.

Le, Phong (2012). Learning semantic parsing. Master's thesis, University of Amsterdam.

Le, Phong and Zuidema, Willem (2012). Learning compositional semantics for open domain semantic parsing. In *Proceedings of COLING 2012*, page 33–50.

Lewis, Mike, Lee, Kenton, and Zettlemoyer, Luke (2016). LSTM CCG parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231.

Lewis, Mike and Steedman, Mark (2014). A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, Doha, Qatar.

Liang, Percy, Jordan, Michael, and Klein, Dan (2011). Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599. Association for Computational Linguistics.

Lindop, Jeremy and Tsujii, Jun-ichi (1991). Complex transfer in MT: A survey of examples. Technical Report 234, Center for Computational Linguistics, New Haven, CT.

Ljunglöf, Peter (2014). Shrdlite: Semantic parsing using a handmade grammar. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 556–559.

Lu, Wei, Ng, Hwee Tou, Lee, Wee Sun, and Zettlemoyer, Luke S. (2008). A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 783–792.

Manshadi, Mehdi and Allen, James (2011). Unrestricted quantifier scope disambiguation. In *Proceedings of TextGraphs-6: Graph-based*

Methods for Natural Language Processing, pages 51–59, Portland, Oregon.

Marcus, Mitchell P., Santorini, Beatrice, and Marcinkiewicz, Mary Ann (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Martins, Ronaldo (2012). Le petit prince in UNL. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*.

Mattelaer, Willem, Verbeke, Mathias, and Nitti, Davide (2014). KUL-Eval: A combinatory categorial grammar approach for improving semantic parsing of robot commands using spatial context. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 385–390.

Matuszek, Cynthia, Fox, Dieter, and Koscher, Karl (2010). Following directions using statistical machine translation. In *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction*, pages 251–258.

May, Jonathan (2016). Semeval-2016 task 8: Meaning representation parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1063–1073.

McDonald, Ryan, Hall, Keith, and Mann, Gideon (2010). Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 456–464.

McDonald, Ryan, Petrov, Slav, and Hall, Keith (2011). Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 62–72.

Miller, Scott, Bobrow, Robert, Ingria, Robert, and Schwartz, Richard (1994). Hidden understanding models of natural language. In *Pro-*

ceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, pages 25–32.

Minnen, Guido, Carroll, John, and Pearce, Darren (2001). Applied morphological processing of English. *Journal of Natural Language Engineering*, 7(3):207–223.

Montague, Richard (1970). Universal grammar. *Theoria*, 36:373–398.

Montague, Richard (1973). The proper treatment of quantification in ordinary English. In Hintikka, Jaakko, Moravcsik, Julius M.E., and Suppes, Patrick, editors, *Approaches to Natural Language*, pages 221–242. Reidel, Dordrecht.

Muskens, Reinhard (1996). Combining Montague Semantics and Discourse Representation. *Linguistics and Philosophy*, 19:143–186.

Naseem, Tahira, Barzilay, Regina, and Globerson, Amir (2012). Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637.

Nguyen, Le-Minh, Shimazu, Akira, and Phan, Xuan-Hieu (2006). Semantic parsing with structured SVM ensemble classification models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 619–626.

Nivre, Joakim (2003). An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Och, Franz Josef and Ney, Hermann (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, Franz Josef and Ney, Hermann (2004). The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Oepen, Stephan, Toutanova, Kristina, Shieber, Stuart, Manning, Christopher, Flickinger, Dan, and Brants, Thorsten (2002). The LinGO Redwoods treebank: Motivation and preliminary applications. In *COLING 2002: The 19th International Conference on Computational Linguistics*, pages 1253–1257.

Ovchinnikova, Ekaterina (2012). *Integration of World Knowledge for Natural Language Understanding*. Atlantis Press, Springer.

Packard, Woodley (2014). UW-MRS: Leveraging a deep grammar for robotic spatial commands. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 812–816.

Palmer, Martha, Kingsbury, Paul, and Gildea, Daniel (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Papineni, Kishore A., Roukos, Salim, and Ward, Todd R. (1997). Feature-based language understanding. In *Fifth European Conference on Speech Communication and Technology*.

Petrov, Slav and Klein, Dan (2007). Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, pages 404–411.

Pieraccini, Roberto, Tzoukermann, Evelyne, Gorelov, Zahkhar, Gauvain, Jean-Luc, Levin, Esther, Lee, Chin-Hui, and Wilpon, Jay G. (1992). A speech understanding system based on statistical representation of semantics. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 193–196.

Poon, Hoifung (2013). Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 933–943.

Poon, Hoifung and Domingos, Pedro (2009). Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10.

Price, Patti J. (1990). Evaluation of spoken language systems: the ATIS domain. In *Proceedings of the Third DARPA Speech and Natural Language Workshop*, pages 91–95.

Raymond, Eric S. (1999). *The Cathedral and the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media, Sebastopol, California.

Reddy, Siva, Lapata, Mirella, and Steedman, Mark (2014). Large-scale semantic parsing without question-answer pairs. *Transactions of the Association of Computational Linguistics*, 2:377–392.

Reyle, Uwe (1993). Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics*, 10:123–179.

Rosetta, M. T. (1994). *Compositional Translation*. Kluwer Academic Publishers.

Ross, John Robert (1967). *Constraints on variables in syntax*. PhD thesis, MIT.

Sag, Ivan A., Baldwin, Timothy, Bond, Francis, Copestake, Ann, and Flickinger, Dan (2002). Multiword expressions: A pain in the neck for NLP. In Gelbukh, Alexander, editor, *Computational Linguistics and Intelligent Text Processing*, pages 1–15. Springer.

Schmid, Helmut (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*.

Schmid, Helmut (1995). Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*.

Sekine, Satoshi, Sudo, Kiyoshi, and Nobata, Chikashi (2002). Extended named entity hierarchy. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*.

Silla Jr., Carlos N. and Kaestner, Celso A. A. (2004). An analysis of sentence boundary detection systems for english and portuguese documents. In *Fifth International Conference on Intelligent Text Processing and Computational Linguistics*, volume 2945 of *Lectures Notes in Computer Science*, pages 135–141. Springer.

Simons, Mandy, Tonhauser, Judith, Beaver, David, and Roberts, Craige (2010). What projects and why. *Semantics and Linguistic Theory*, 20(0):309–327.

Srinivas, Bangalore and Joshi, Aravind (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Srinivasan, Prakash and Yates, Alexander (2009). Quantifier scope disambiguation using extracted pragmatic knowledge: Preliminary results. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1465–1474.

Steedman, Mark (2001). *The Syntactic Process*. The MIT Press.

Steedman, Mark and Baldridge, Jason (2011). Combinatory categorial grammar. In Borsley, Robert and Borjars, Kersti, editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar*. Wiley-Blackwell.

Steedman, Mark J. (2012). *Taking scope: the natural semantics of quantifiers*. The MIT Press, Cambridge, MA.

Stoyanchev, Svetlana, Jung, Hyuckchul, Chen, John, and Bangalore, Srnivas (2014). AT&T: The tag&parse approach to semantic parsing of robot spatial commands. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 109–113.

Sulger, Sebastian, Butt, Miriam, King, Tracy Holloway, Meurer, Paul, Laczkó, Tibor, Rákosi, György, Dione, Cheikh Bamba, Dyvik, Helge, Rosén, Victoria, Smedt, Koenraad De, Patejuk, Agnieszka, Özlem Çetinoğlu, Arka, I Wayan, and Mistica, Meladel (2013). ParGramBank: The ParGram parallel treebank. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 550–560.

Täckström, Oscar, McDonald, Ryan, and Nivre, Joakim (2013). Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071.

Tanaka, Yasuhito (2001). Compilation of a multilingual parallel corpus. *Proceedings of PACLING 2001*, pages 265–268.

Tang, Lappoon R. and Mooney, Raymond J. (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, pages 466–477.

Thompson, Cynthia A. and Mooney, Raymond J. (2003). Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research*, 18:1–44.

Tiedemann, Jörg (2014). Rediscovering annotation projection for cross-lingual parser induction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1854–1864.

Tiedemann, Jörg, Agić, Željko, and Nivre, Joakim (2014). Treebank translation for cross-lingual parser induction. In *Eighteenth Conference on Computational Natural Language Learning*.

Titov, Ivan and Klementiev, Alexandre (2011). A Bayesian model for unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 1445–1455.

Tonhauser, Judith, Beaver, David, Roberts, Craige, and Simons, Mandy (2013). Toward a taxonomy of projective content. *Language*, 89(1):66–109.

Van der Sandt, Rob A. (1992). Presupposition projection as anaphora resolution. *Journal of Semantics*, 9:333–377.

Venhuizen, Noortje, Basile, Valerio, Evang, Kilian, and Bos, Johan (2013a). Gamification for word sense labeling. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*.

Venhuizen, Noortje J., Bos, Johan, and Brouwer, Harm (2013b). Parsimonious semantic representations with projection pointers. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 252–263, Potsdam, Germany. Association for Computational Linguistics.

Venhuizen, Noortje Joost, Bos, Johan, Hendriks, Petra, and Brouwer, Harm (2014). How and why conventional implicatures project. In Snider, Todd, Wiegand, Jennifer, and D'Antonio, Sarah, editors, *Proceedings of Semantics and Linguistic Theory (SALT)*, volume 24, pages 63–83, New York.

Vijay-Shanker, Krishnamurti and Weir, David J (1994). The equivalence of four extensions of context-free grammars. *Mathematical systems theory*, 27(6):511–546.

Villavicencio, Aline (1997). Building a wide-coverage combinatory categorial grammar. Master's thesis, University of Cambridge.

Vlachos, Andreas and Clark, Stephen (2014). A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics*, 2:547–559.

Wang, Adrienne, Kwiatkowski, Tom, and Zettlemoyer, Luke (2014). Morpho-syntactic lexical generalization for CCG semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1284–1295.

Wang, Chuan, Xue, Nianwen, and Pradhan, Sameer (2015a). Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 857–862.

Wang, Chuan, Xue, Nianwen, and Pradhan, Sameer (2015b). A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375.

Weck, Benno (2015). Assessing the impact of manual corrections in the Groningen Meaning Bank. Master's thesis, University of Groningen.

Weiss, David, Alberti, Chris, Collins, Michael, and Petrov, Slav (2015). Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333.

Werling, Keenon, Angeli, Gabor, and Manning, Christopher D. (2015). Robust subgraph generation improves abstract meaning representation parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 982–991.

Wong, Yuk Wuh and Mooney, Raymond (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967.

Wong, Yuk Wah and Mooney, Raymond J. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446.

Xu, Wenduan, Auli, Michael, and Clark, Stephen (2016). Expected F-measure training for shift-reduce parsing with recurrent neural networks. In *Proceedings of NAACL-HLT 2016*, pages 210–220.

Xu, Wenduan, Clark, Stephen, and Zhang, Yue (2014). Shift-reduce CCG parsing with a dependency model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 218–227.

Yao, Xuchen, Berant, Jonathan, and van Durme, Benjamin (2014). Freebase QA: Information extraction and semantic parsing? In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 82–86.

Yao, Xuchen and van Durme, Benjamin (2014). Information extraction over structured data: Question answering with Freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966.

Zaenen, Annie, Carletta, Jean, Garretson, Gregory, Bresnan, Joan, Koontz-Garboden, Andrew, Nikitina, Tatiana, O'Connor, M Catherine, and Wasow, Tom (2004). Animacy encoding in english: why and how. In *Proceedings of the 2004 ACL Workshop on Discourse Annotation*, pages 118–125. Association for Computational Linguistics.

Zelle, John M. and Mooney, Raymond J. (1996). Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the Eleventh National Conference of the American Association for Artificial Intelligence (AAAI-93)*, pages 817–822.

Zeman, Daniel and Resnik, Philip (2008). Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pages 35–42.

Zettlemoyer, Luke and Collins, Michael (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 658–666. AUAI Press.

Zettlemoyer, Luke S. and Collins, Michael (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687.

Zettlemoyer, Luke S. and Collins, Michael (2009). Learning context-dependent mappings from sentences to logical form. In *Proceedings of*

the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 976–984.

Zhang, Yue and Clark, Stephen (2011). Shift-reduce CCG parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 683–692.

Zhang, Yue and Nivre, Joakim (2011). Transition-based parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Zhao, Kai and Huang, Liang (2015). Type-driven incremental semantic parsing with polymorphism. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Zhu, Muhua, Zhang, Yue, Chen, Wenliang, Zhang, Min, and Zhu, Jingbo (2013). Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443.

Groningen Dissertations in Linguistics (GRODIL)

1. Henriëtte de Swart (1991). *Adverbs of Quantification: A Generalized Quantifier Approach.*
2. Eric Hoekstra (1991). *Licensing Conditions on Phrase Structure.*
3. Dicky Gilbers (1992). *Phonological Networks. A Theory of Segment Representation.*
4. Helen de Hoop (1992). *Case Configuration and Noun Phrase Interpretation.*
5. Gosse Bouma (1993). *Nonmonotonicity and Categorial Unification Grammar.*
6. Peter Blok (1993). *The Interpretation of Focus: an epistemic approach to pragmatics.*
7. Roelien Bastiaanse (1993). *Studies in Aphasia.*
8. Bert Bos (1993). *Rapid User Interface Development with the Script Language Gist.*
9. Wim Kosmeijer (1993). *Barriers and Licensing.*
10. Jan-Wouter Zwart (1993). *Dutch Syntax: A Minimalist Approach.*
11. Mark Kas (1993). *Essays on Boolean Functions and Negative Polarity.*
12. Ton van der Wouden (1994). *Negative Contexts.*
13. Joop Houtman (1994). *Coordination and Constituency: A Study in Categorial Grammar.*
14. Petra Hendriks (1995). *Comparatives and Categorial Grammar.*
15. Maarten de Wind (1995). *Inversion in French.*
16. Jelly Julia de Jong (1996). *The Case of Bound Pronouns in Peripheral Romance.*
17. Sjoukje van der Wal (1996). *Negative Polarity Items and Negation: Tandem Acquisition.*
18. Anastasia Giannakidou (1997). *The Landscape of Polarity Items.*
19. Karen Lattewitz (1997). *Adjacency in Dutch and German.*
20. Edith Kaan (1997). *Processing Subject-Object Ambiguities in Dutch.*
21. Henny Klein (1997). *Adverbs of Degree in Dutch.*
22. Leonie Bosveld-de Smet (1998). *On Mass and Plural Quantification: The Case of French 'des'/'du'-NPs.*
23. Rita Landeweerd (1998). *Discourse Semantics of Perspective and Temporal Structure.*
24. Mettina Veenstra (1998). *Formalizing the Minimalist Program.*
25. Roel Jonkers (1998). *Comprehension and Production of Verbs in Aphasic Speakers.*

26. Erik F. Tjong Kim Sang (1998). *Machine Learning of Phonotactics.*
27. Paulien Rijkhoek (1998). *On Degree Phrases and Result Clauses.*
28. Jan de Jong (1999). *Specific Language Impairment in Dutch: Inflectional Morphology and Argument Structure.*
29. Hae-Kyung Wee (1999). *Definite Focus.*
30. Eun-Hee Lee (2000). *Dynamic and Stative Information in Temporal Reasoning: Korean Tense and Aspect in Discourse.*
31. Ivilin Stoianov (2001). *Connectionist Lexical Processing.*
32. Klarien van der Linde (2001). *Sonority Substitutions.*
33. Monique Lamers (2001). *Sentence Processing: Using Syntactic, Semantic, and Thematic Information.*
34. Shalom Zuckerman (2001). *The Acquisition of "Optional" Movement.*
35. Rob Koeling (2001). *Dialogue-Based Disambiguation: Using Dialogue Status to Improve Speech Understanding.*
36. Esther Ruigendijk (2002). *Case Assignment in Agrammatism: a Cross-linguistic Study.*
37. Tony Mullen (2002). *An Investigation into Compositional Features and Feature Merging for Maximum Entropy-Based Parse Selection.*
38. Nanette Bienfait (2002). *Grammatica-onderwijs aan allochtone jongeren.*
39. Dirk-Bart den Ouden (2002). *Phonology in Aphasia: Syllables and Segments in Level-specific Deficits.*
40. Rienk Withaar (2002). *The Role of the Phonological Loop in Sentence Comprehension.*
41. Kim Sauter (2002). *Transfer and Access to Universal Grammar in Adult Second Language Acquisition.*
42. Laura Sabourin (2003). *Grammatical Gender and Second Language Processing: An ERP Study.*
43. Hein van Schie (2003). *Visual Semantics.*
44. Lilia Schürcks-Grozeva (2003). *Binding and Bulgarian.*
45. Stasinos Konstantopoulos (2003). *Using ILP to Learn Local Linguistic Structures.*
46. Wilbert Heeringa (2004). *Measuring Dialect Pronunciation Differences using Levenshtein Distance.*
47. Wouter Jansen (2004). *Laryngeal Contrast and Phonetic Voicing: A Laboratory Phonology Approach to English, Hungarian and Dutch.*
48. Judith Rispens (2004). *Syntactic and Phonological Processing in Developmental Dyslexia.*
49. Danielle Bougaïré (2004). *L'approche communicative des campagnes de sensibilisation en santé publique au Burkina Faso: les cas de la planification familiale, du sida et de l'excision.*
50. Tanja Gaustad (2004). *Linguistic Knowledge and Word Sense Disambiguation.*
51. Susanne Schoof (2004). *An HPSG Account of Nonfinite Verbal Complements in Latin.*
52. M. Begoña Villada Moirón (2005). *Data-driven identification of fixed expressions and their modifiability.*
53. Robbert Prins (2005). *Finite-State Pre-Processing for Natural Language Analysis.*
54. Leonoor van der Beek (2005). *Topics in Corpus-Based Dutch Syntax.*

55. Keiko Yoshioka (2005). *Linguistic and gestural introduction and tracking of referents in L1 and L2 discourse.*
56. Sible Andringa (2005). *Form-focused instruction and the development of second language proficiency.*
57. Joanneke Prenger (2005). *Taal telt! Een onderzoek naar de rol van taalvaardigheid en tekstbegrip in het realistisch wiskundeonderwijs.*
58. Neslihan Kansu-Yetkiner (2006). *Blood, Shame and Fear: Self-Presentation Strategies of Turkish Women's Talk about their Health and Sexuality.*
59. Mónika Z. Zempléni (2006). *Functional imaging of the hemispheric contribution to language processing.*
60. Maartje Schreuder (2006). *Prosodic Processes in Language and Music.*
61. Hidetoshi Shiraishi (2006). *Topics in Nivkh Phonology.*
62. Tamás Biró (2006). *Finding the Right Words: Implementing Optimality Theory with Simulated Annealing.*
63. Dieuwke de Goede (2006). *Verbs in Spoken Sentence Processing: Unraveling the Activation Pattern of the Matrix Verb.*
64. Eleonora Rossi (2007). *Clitic production in Italian agrammatism.*
65. Holger Hopp (2007). *Ultimate Attainment at the Interfaces in Second Language Acquisition: Grammar and Processing.*
66. Gerlof Bouma (2008). *Starting a Sentence in Dutch: A corpus study of subject- and object-fronting.*
67. Julia Klitsch (2008). *Open your eyes and listen carefully. Auditory and audiovisual speech perception and the McGurk effect in Dutch speakers with and without aphasia.*
68. Janneke ter Beek (2008). *Restructuring and Infinitival Complements in Dutch.*
69. Jori Mur (2008). *Off-line Answer Extraction for Question Answering.*
70. Lonneke van der Plas (2008). *Automatic Lexico-Semantic Acquisition for Question Answering.*
71. Arjen Versloot (2008). *Mechanisms of Language Change: Vowel reduction in 15th century West Frisian.*
72. Ismail Fahmi (2009). *Automatic term and Relation Extraction for Medical Question Answering System.*
73. Tuba Yarbay Duman (2009). *Turkish Agrammatic Aphasia: Word Order, Time Reference and Case.*
74. Maria Trofimova (2009). *Case Assignment by Prepositions in Russian Aphasia.*
75. Rasmus Steinkrauss (2009). *Frequency and Function in WH Question Acquisition. A Usage-Based Case Study of German L1 Acquisition.*
76. Marjolein Deunk (2009). *Discourse Practices in Preschool. Young Children's Participation in Everyday Classroom Activities.*
77. Sake Jager (2009). *Towards ICT-Integrated Language Learning: Developing an Implementation Framework in terms of Pedagogy, Technology and Environment.*
78. Francisco Dellatorre Borges (2010). *Parse Selection with Support Vector Machines.*
79. Geoffrey Andogah (2010). *Geographically Constrained Information Retrieval.*
80. Jacqueline van Kruiningen (2010). *Onderwijsontwerp als conversatie. Probleemoplossing in interprofessioneel overleg.*

81. Robert G. Shackleton (2010). *Quantitative Assessment of English-American Speech Relationships.*
82. Tim Van de Cruys (2010). *Mining for Meaning: The Extraction of Lexico-semantic Knowledge from Text.*
83. Therese Leinonen (2010). *An Acoustic Analysis of Vowel Pronunciation in Swedish Dialects.*
84. Erik-Jan Smits (2010). *Acquiring Quantification. How Children Use Semantics and Pragmatics to Constrain Meaning.*
85. Tal Caspi (2010). *A Dynamic Perspective on Second Language Development.*
86. Teodora Mehotcheva (2010). *After the fiesta is over. Foreign language attrition of Spanish in Dutch and German Erasmus Students.*
87. Xiaoyan Xu (2010). *English language attrition and retention in Chinese and Dutch university students.*
88. Jelena Prokić (2010). *Families and Resemblances.*
89. Radek Šimík (2011). *Modal existential wh-constructions.*
90. Katrien Colman (2011). *Behavioral and neuroimaging studies on language processing in Dutch speakers with Parkinson's disease.*
91. Siti Mina Tamah (2011). *A Study on Student Interaction in the Implementation of the Jigsaw Technique in Language Teaching.*
92. Aletta Kwant (2011). *Geraakt door prentenboeken. Effecten van het gebruik van prentenboeken op de sociaal-emotionele ontwikkeling van kleuters.*
93. Marlies Kluck (2011). *Sentence amalgamation.*
94. Anja Schüppert (2011). *Origin of asymmetry: Mutual intelligibility of spoken Danish and Swedish.*
95. Peter Nabende (2011). *Applying Dynamic Bayesian Networks in Transliteration Detection and Generation.*
96. Barbara Plank (2011). *Domain Adaptation for Parsing.*
97. Çağrı Çöltekin (2011). *Catching Words in a Stream of Speech: Computational simulations of segmenting transcribed child-directed speech.*
98. Dörte Hessler (2011). *Audiovisual Processing in Aphasic and Non-Brain-Damaged Listeners: The Whole is More than the Sum of its Parts.* Herman Heringa (2012). *Appositional constructions.*
99. Herman Heringa (2012). *Appositional constructions.*
100. Diana Dimitrova (2012). *Neural Correlates of Prosody and Information Structure.*
101. Harwintha Anjarningsih (2012). *Time Reference in Standard Indonesian Agrammatic Aphasia.*
102. Myrte Gosen (2012). *Tracing learning in interaction. An analysis of shared reading of picture books at kindergarten.*
103. Martijn Wieling (2012). *A Quantitative Approach to Social and Geographical Dialect Variation.*
104. Gisi Cannizzaro (2012). *Early word order and animacy.*
105. Kostadin Cholakov (2012). *Lexical Acquisition for Computational Grammars. A Unified Model.*
106. Karin Beijering (2012). *Expressions of epistemic modality in Mainland Scandinavian.*

A study into the lexicalization-grammaticalization-pragmaticalization interface.

107. Veerle Baaijen (2012). *The development of understanding through writing.*
108. Jacolien van Rij (2012). *Pronoun processing: Computational, behavioral, and psychophysiological studies in children and adults.*
109. Ankelien Schippers (2012). *Variation and change in Germanic long-distance dependencies.*
110. Hanneke Loerts (2012). *Uncommon gender: Eyes and brains, native and second language learners, & grammatical gender.*
111. Marjoleine Sloos (2013). *Frequency and phonological grammar: An integrated approach. Evidence from German, Indonesian, and Japanese.*
112. Aysa Arylova (2013). *Possession in the Russian clause. Towards dynamicity in syntax.*
113. Daniël de Kok (2013). *Reversible Stochastic Attribute-Value Grammars.*
114. Gideon Kotzé (2013). *Complementary approaches to tree alignment: Combining statistical and rule-based methods.*
115. Fridah Katushemererwe (2013). *Computational Morphology and Bantu Language Learning: an Implementation for Runyakitara.*
116. Ryan C. Taylor (2013). *Tracking Referents: Markedness, World Knowledge and Pronoun Resolution.*
117. Hana Smiskova-Gustafsson (2013). *Chunks in L2 Development: A Usage-Based Perspective.*
118. Milada Walková (2013). *The aspectual function of particles in phrasal verbs.*
119. Tom O. Abuom (2013). *Verb and Word Order Deficits in Swahili-English bilingual agrammatic speakers.*
120. Gülsen Yılmaz (2013). *Bilingual Language Development among the First Generation Turkish Immigrants in the Netherlands.*
121. Trevor Benjamin (2013). *Signaling Trouble: On the linguistic design of other-initiation of repair in English Conversation.*
122. Nguyen Hong Thi Phuong (2013). *A Dynamic Usage-based Approach to Second Language Teaching.*
123. Harm Brouwer (2014). *The Electrophysiology of Language Comprehension: A Neurocomputational Model.*
124. Kendall Decker (2014). *Orthography Development for Creole Languages.*
125. Laura S. Bos (2015). *The Brain, Verbs, and the Past: Neurolinguistic Studies on Time Reference.*
126. Rimke Groenewold (2015). *Direct and indirect speech in aphasia: Studies of spoken discourse production and comprehension.*
127. Huiping Chan (2015). *A Dynamic Approach to the Development of Lexicon and Syntax in a Second Language.*
128. James Griffiths (2015). *On appositives.*
129. Pavel Rudnev (2015). *Dependency and discourse-configurationality: A study of Avar.*
130. Kirsten Kolstrup (2015). *Opportunities to speak. A qualitative study of a second language in use.*
131. Güliz Güneş (2015). *Deriving Prosodic structures.*
132. Cornelia Lahmann (2015). *Beyond barriers. Complexity, accuracy, and fluency in*

long-term L2 speakers' speech.

133. Sri Wachyunni (2015). *Scaffolding and Cooperative Learning: Effects on Reading Comprehension and Vocabulary Knowledge in English as a Foreign Language.*
134. Albert Walsweer (2015). *Ruimte voor leren. Een etnografisch onderzoek naar het verloop van een interventie gericht op versterking van het taalgebruik in een knowledge building environment op kleine Friese basisscholen.*
135. Aleyda Lizeth Linares Calix (2015). *Raising Metacognitive Genre Awareness in L2 Academic Readers and Writers.*
136. Fathima Mufeeda Irshad (2015). *Second Language Development through the Lens of a Dynamic Usage-Based Approach.*
137. Oscar Strik (2015). *Modelling analogical change. A history of Swedish and Frisian verb inflection.*
138. He Sun (2015). *Predictors and stages of very young child EFL learners' English development in China.*
139. Marieke Haan (2015). *Mode Matters. Effects of survey modes on participation and answering behavior.*
140. Nienke Houtzager (2015). *Bilingual advantages in middle-aged and elderly populations.*
141. Noortje Joost Venhuizen (2015). *Projection in Discourse: A data-driven formal semantic analysis.*
142. Valerio Basile (2015). *From Logic to Language: Natural Language Generation from Logical Forms.*
143. Jinxing Yue (2016). *Tone-word Recognition in Mandarin Chinese: Influences of lexical-level representations.*
144. Seçkin Arslan (2016). *Neurolinguistic and Psycholinguistic Investigations on Evidentiality in Turkish.*
145. Rui Qin (2016). *Neurophysiological Studies of Reading Fluency. Towards Visual and Auditory Markers of Developmental Dyslexia.*
146. Kashmiri Stec (2016). *Visible Quotation: The Multimodal Expression of Viewpoint.*
147. Yinxing Jin (2016). *Foreign language classroom anxiety: A study of Chinese university students of Japanese and English over time.*
148. Joost Hurkmans (2016). *The Treatment of Apraxia of Speech. Speech and Music Therapy, an Innovative Joint Effort*
149. Franziska Köder (2016). *Between direct and indirect speech: The acquisition of pronouns in reported speech.*
150. Femke Swarte (2016). *Predicting the mutual intelligibility of Germanic languages from linguistic and extra-linguistic factors.*
151. Sanne Kuijper (2016). *Communication abilities of children with ASD and ADHD. Production, comprehension, and cognitive mechanisms.*
152. Jelena Golubović (2016). *Mutual intelligibility in the Slavic language area.*
153. Nynke van der Schaaf (2016). *Kijk eens wat ik kan! Sociale praktijken in interactie tussen kinderen van 4-8 jaar in de buitenschoolse opvang.*
154. Simon Šuster (2016). *Empirical studies on word representations.*
155. Kilian Evang (2016). *Cross-lingual Semantic Parsing with Categorial Grammars.*

GRODIL
Secretary of the Department of General Linguistics
Postbus 716
9700 AS Groningen
The Netherlands