Best-Fit Constructional Analysis

John Edward Bryant

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-100
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-100.html

August 20, 2008

Best-Fit Constructional Analysis

by

John Edward Bryant

B.A. (University of California at Berkeley) 2000
M.S. (University of California at Berkeley) 2003

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:
Professor Jerome Feldman, Chair
Professor George Lakoff
Professor Dan Klein

Fall 2008

The dissertation of John Edward Bryant is approved:

Chair Date

Date

Date

University of California, Berkeley

Fall 2008

Best-Fit Constructional Analysis

Abstract

Best-Fit Constructional Analysis

by

John Edward Bryant
Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Jerome Feldman, Chair

This dissertation describes a process called *best-fit constructional analysis* and the associated implementation called the *Constructional Analyzer*. The constructional analyzer takes an utterance as input and performs deep semantic analysis, mapping the utterance onto its most likely interpretation. The best-fit constructional analyzer is unique because it combines *Embodied Construction Grammar* with the power of *best-fit* processing. This combination enables the constructional analyzer to be both a cognitive model of interpretation and a practical semantic analysis system.

The best-fit constructional analyzer performs incremental unification grammar parsing, using a factored probabilistic model over syntax and semantics to guide interpretation. The constructional analyzer has been applied to a range of applications: a) It is a tool for building and testing construction grammars. b) It is a psychologically plausible model of human interpretation that makes predictions about reading time that match experimental evidence. c) It is a practical system for semantic analysis that has been tested on a corpus of Mandarin child-parent dialogues.

Professor Jerome Feldman
Dissertation Committee Chair

To my parents

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank everyone that put up with me through the years:

- My parents, sister and grandparents have had to put up with me for the longest out of anyone.
- My friends Jeremy, Jason, Sean and Michael have had to put up with me since undergrad.
- My friends and (former) housemates Marie, Joe, Christine and Brian have had to put up with me at home.
- My friends and colleagues Eva, Nancy, Joe, Steve, Michael, Josef, Ellen and Leon have had to put up with me at work.
- My advisor Jerry has *really* had to put up with me.
- My secondary advisor Srini has also *really* had to put up with me, but not as long as Jerry has *really* had to put up with me.
- My committee members George and Dan have had to put up with me during the process of writing my dissertation.
- My best girl Angela has had to put up with me and my grumpiness during the dregs of this process.

Chapter 1

Introduction

This dissertation describes a process called *best-fit constructional analysis* and the associated implementation called the *Constructional Analyzer* (or often just "the analyzer" for short). The constructional analyzer takes an utterance as input and performs deep semantic analysis, mapping the input utterance onto its most likely interpretation. The best-fit constructional analyzer is unique because it combines *Embodied Construction Grammar* with the power of *best-fit* processing. This combination enables the best-fit constructional analyzer to be both a cognitive model of interpretation and a practical domain-specific semantic analysis system.

Construction grammar [21] [37] [38] is a theory of grammar that states that the rules of a language are mappings between form constraints and meaning constraints. Each mapping is called a *construction*. Figure 1.1 provides an intuitive introduction to some example constructions. Constructions are a powerful tool for semantic analysis because they are explicit about the relation between form and meaning. Parsing an utterance with a construction grammar results in a set of well-formed semantic constraints describing the utterance, and thus parsing with construction grammar is equivalent to performing semantic analysis on an utterance. So instead of just calling this process "parsing" or "semantic analysis", we call it *constructional analysis* [5].

Best-fit is a term we use to describe any decision-making process that combines information from multiple domains in a quantitative way. Thus Best-fit constructional analysis is a process in which decisions about how to interpret an utterance (using

constructions) are conditioned on semantic, syntactic and contextual information. In the implemented constructional analyzer, semantic, syntactic and contextual information is combined within a probabilistic model, and the probabilistic model is used to define a ranking function over (partial) interpretations as each word of an input utterance is processed. We refer to this probabilistic model as the *best-fit metric*.

Figure 1.2 provides a schematic view of the analyzer. It takes an utterance, a grammar, and a model of context as input. The grammar is a construction grammar, and therefore contains both syntactic and semantic information. The context model is a structured (relational) representation of discourse and situational context that tracks salient entities. When an utterance is processed, the syntactic, semantic, and contextual constraints in the grammar help determine likely interpretations of the utterance, and references to context (such as pronouns) are resolved according to the context model.

When the analyzer interprets an utterance, it returns what we call an *analysis*. An analysis specifies the constructions that were used to process the utterance and the semantic content of the utterance (the interpretation). The semantic content is called a *Semantic Specification* or semspec, and it consists of an inter-connected set of *frames*, semantic *schemas*, and participants that describes both the semantic content of the utterance and how that semantic content was communicated. A frame is a gestalt representation of some event or scene along with that scene's associated *role*s [20]. Like frames, semantic schemas are also gestalt representations of concepts, except schemas represent neurally motivated embodied primitives such as image schemas [82] and x-schemas [54]. Figure 1.3 explains how a simplified example semspec for the sentence *Joe walked into the house* can be represented in feature structure notation.

1.1 A Practical System and a Cognitive Model

The best-fit constructional analyzer is both a flexible system for semantic analysis and a cognitive model of interpretation. The fields of cognitive/functional linguistics, natural language processing, and psycholinguistics all inform the design of the system.

Example Constructions

Name	Form	Meaning
Suffix -ed	"-ed"	before speech time, completed
Lexical Give	"give"	a give action
Double Object	NP_0 V NP_1 NP_2	Transfer Scene + bindings
WXDY	What is NP_0 doing PP_0	How come NP_0 is PP_0

Figure 1.1: Four example English constructions represented in an intuitive way. The three columns in the Example Constructions table show the name of the construction, a description of its form and a description of its meaning. For example, The Suffix -ed construction is a morphological construction that links the suffix *ed* with the verbal semantics associated with simple past tense. The Lexical Give construction shows that a word like *give* is linked to some kind of representation of a giving action.

The Double Object construction is a sentence construction that associates the double object form with both a notion of Transfer and a set of constraints that link the constituents of the construction to the various players in a Transfer scene. For example, in the sentence *she gave him the book*, the subject NP (she) is the *giver*, the second NP (he) is the *recipient*, and the third NP (the book) is the *gift*.

WXDY is a shorthand for the *What is X doing Y?* construction described by Kay and Fillmore [38]. An example of this construction is the sentence *What's a nice girl like you doing in a place like this?*. As Kay and Fillmore point out, this question is not asking about what activity that the nice girl is doing, but rather has a paraphrase more like *How come a nice girl like you is in a place like this?* which suggests that the situation is inappropriate for some reason. Notice that the WXDY construction leads to an ambiguity in the grammar. This ambiguity motivates the humor in the joke *Waiter! Waiter! What's this fly doing in my soup?*

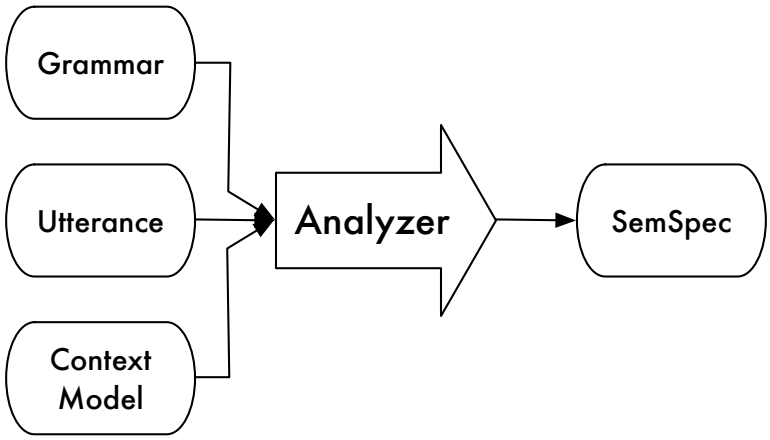

Figure 1.2: A blackbox diagram describing the input and output of the analyzer.

Each field focuses on a different aspect of the parsing/interpretation problem, and as a consequence, each of the three fields have valuable, but very different insights into how humans interpret language. One goal of this dissertation is to unify insights from each of these fields because doing so would provide a single unified cognitive framework for modeling and applying natural language interpretation.

Later chapters provide a much more thorough discussion about how the fields of cognitive linguistics, natural language processing and psycholinguistics influence the design choices of the system. But in order to contrast the constructional analyzer from its related work, related work and ideas from each field are sketched out below:

- Cognitive and functional linguists study the richness of human language. They examine how form and semantic function are related by highlighting and decomposing the semantic complexity found in common everyday language. The idea of construction grammar comes out of this tradition as well as frames, semantic schemas, and the study of productive metaphor [44]. Cognitive linguists do not generally concentrate on computational models of interpretation though, focusing instead on qualitative explanation of how people infer meaning from language. Feldman provides one such qualitative model of interpretation,

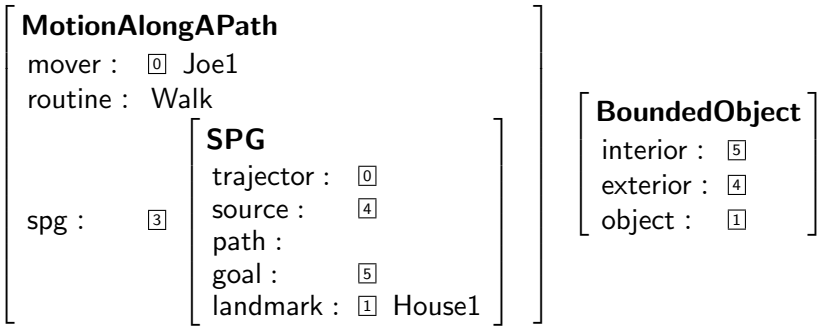

Figure 1.3: A simple semspec for *Joe walked into the house* represented in feature structure notation. A *feature structure* [35] is a typed set of feature-value pairs. The semantics of Joe moving into the house by walking is represented by the frame MotionAlongAPath represented as a feature structure with mover, routine and spg roles. The mover role is the entity moving along the path, the routine role represents the method by which the mover moves along the path, and the spg role refers the path itself.

The semspec contains two semantic schemas SPG and BoundedObject also represented as feature structures. The SPG schema represents a path using roles for the source location, path and goal location. The SPG's trajector role represents the entity construed as moving along the path, and the landmark role represents the location to which the trajector's position is compared. The BoundedObject schema represents an object with an interior and exterior.

Roles can be filled by atomic values such as the mover role being filled by Joe1 or the routine role being filled by Walk as shown in the example, or by other feature structures such as the SPG filling the MotionAlongAPath's spg role.

In addition to being filled by values, a set of features can also be co-indexed. Co-indexed features all share the same filler. In a feature structure, co-indexation is represented using the numbered box notation. For example, the MotionAlongAPath's mover role and the SPG's trajector role are co-indexed because both roles are marked with the $\boxed{0}$. Co-indexation $\boxed{1}$ specifies that House1 is both the landmark of the SPG and the object of the BoundedObject schema meaning that House1 is being thought of as a BoundedObject in this sentence, and that the interior and exterior roles of BoundedObject actually refer to the interior and exterior of House1.

defining language interpretation in terms of neurally motivated processes [19].

- The natural language processing (NLP) community generally concentrates on broad coverage systems. They show how to infer some of the hidden structure of an utterance in efficient, robust and mathematically well-motivated ways, even when scaling up to huge data sets. However these algorithms are generally not constrained by insights into human processing, and the hidden structures they infer for an utterance are too shallow to be considered an interpretation of the utterance. For example, broad coverage statistical parsers [10], [11], [62] can find a syntactic structure for pretty much any sentence of English, but phrase structure trees are obviously not interpretations. Systems that perform Propbank [61] or FrameNet [22] role assignment [24] [64] provide useful shallow semantic information about a sentence, but again the returned semantic information cannot be considered an interpretation.

  Recent work in the NLP community on semantic analysis has lead to promising results for single domain systems. Zettlemoyer and Collins show how to learn a weighted CCG [7] grammar for the domains of basic geography [90] or travel [91]. Each CCG grammar rule is annotated with a fragment of lambda calculus that approximates the meaning of the rule, and a complete interpretation is a well-formed SQL query. Wong and Mooney show that the semantic analysis problem can be treated as a machine translation problem between natural language and the formal language of logical formulas [89]. They use techniques from machine translation to learn synchronous grammars between the two.

  In between NLP and functional linguistics lies the relatively broad coverage parsers built for linguistic frameworks such as LFG [47] and HPSG [63]. Work by Kaplan et al. [66] uses a disambiguation system on top of the LFG parser to quickly interpret a sentence. Work by Toutanova et al. employs a similar mechanism for HPSG disambiguation [87].

- Psycholinguists investigate the process of human language interpretation in an experimental setting. By discovering the differences between sentences that are

easy to interpret and those that are harder to interpret, psycholinguists infer constraints on how humans process language. The computational models built by psycholinguists make predictions consistent with the experimental data, but tend to focus on simplicity at the expense of structure. Most psycholinguistic models such as [46] [30] therefore do not function as models of interpretation with the exception of Narayanan and Jurafsky [57], Jurafsky [34] and Pado [59]. While these three model cannot provide deep semantic interpretation *and* predict reading time difficulty, all three are extremely influential on the work described here.

No computational model of interpretation takes all three of these fields seriously with the exception of this work. The analyzer can be used for a combined set of tasks that is unmatched by any other system. In summary, the analyzer is:

- a practical, easily-customized semantic analysis system,
- a system capable of analyzing languages besides English and dealing with context-reliant languages like Mandarin,
- a tool for the building and testing construction grammars that is already in use by linguists for pushing the boundaries of construction grammar,
- a psychologically-plausible model of human interpretation that makes predictions about reading time that match experimental evidence.

1.2 Best-Fit

Though it goes by different names in different fields, motivation for best-fit processing comes from many of the fields of cognitive science. In psychology and psycholinguistic sentence processing literature, best-fit processing models are known as *interactionist* or *constraint-based* models. According to McRae, Spivey and Tannenhaus:

> In contrast [to a modular approach], constraint-based models assume that multiple syntactic alternatives are evaluated using both linguistic and non-linguistic sources of constraint. The comprehension system continuously integrates all the relevant and available information in order to compute the interpretation that best satisfies those constraints. [51]

Psycholinguistic models that fit the interactionist paradigm are the aforementioned model of Narayanan and Jurafsky and the connectionist model that McRae et al. propose in [51].

Closely related to the *constraint-based* paradigm are connectionist models and in particular structured connectionist models . Connectionist models are best-fit models that use spreading activation to combine information from multiple domains and competition between units in the connectionist network to model competing hypotheses. Lane and Henderson show how to use a connectionist network to parse utterances into syntactic structures [45], while Feldman provides a framework for reducing language interpretation to connectionist models [19].

Construction grammar itself is a kind of structural best-fit theory of linguistics. Though it is not a quantitative definition, construction grammar defines grammaticality in terms of both formal properties (syntax) and function (semantic and pragmatic constraints). Another early best-fit idea found in the linguistics community is Smolensky's Harmony Theory [76]. Harmony Theory is based on the assumption of symmetric neural connections, but linking harmony theory to widely-used linguistic theories such as optimality theory a problem left as future work.

Natural language processing has been investigating a form of best-fit sentence processing at least since joint models of lexicalized PCFGs were shown to be useful for syntactic parsing [10] [11]. These approaches use lexical dependency information as a proxy for direct semantic information. Less progress has been made on semantic analysis and/or conditioning the parsing process directly on semantic information. Ge and Mooney [23] extend Collin's generative model [11] to generate semantic labels as well as syntactic categories, developing something like a probabilistic form of traditional semantic grammars.

1.2.1 Factored Models

Factored models, as I will be arguing in chapter 4, are a simple, elegant and accurate way to represent best-fit metrics. A factored model decomposes a complex joint distribution over a set of variables v into a set of simpler distributions over disjoint subsets $v_0, v_1 ... v_k$ of v. The likelihood of an assignment over v is approximated by some function f over the distributions of $\Pr(v_0), \Pr(v_1) ... \Pr(v_k)$. More precisely:

$$\Pr(v) \approx f(\Pr(v_0), \Pr(v_1) ... \Pr(v_k))$$

In general, complete probabilistic inference with the complex distribution over v and inference using the factored model do not yield the same result. The accuracy of the inference is sacrificed to improve the speed of the inference. However in many NLP tasks, exact inference is not required. For example, with best-fit constructional analysis, finding the correct interpretation is more important than knowing its exact likelihood.

Because inference in NLP systems has gotten progressively more expensive as the complexity of the systems increase, factored models have recently been shown to be useful for a range of tasks. Klein and Manning [42] employ a model factored into a syntax model and a lexical dependency model for guiding the search for a parse. Narayanan and Jurafsky [57] employ a probabilistic model factored into a syntax model, a thematic role assignment model, and an n-gram model for predicting reading time. Haghighi, DeNero and Klein [27] further explore factored models as a tool for improving the speed and accuracy of various NLP search processes.

1.3 A Neural Theory of Language

Conspicuously left out of the discussion so far is how my work relates to the brain. The work described in this dissertation is at too high a level of abstraction to be considered a connectionist model, but according to the Neural Theory of Language (NTL) [19], models at higher levels of abstraction can still be a kind of neurally-motivated model.

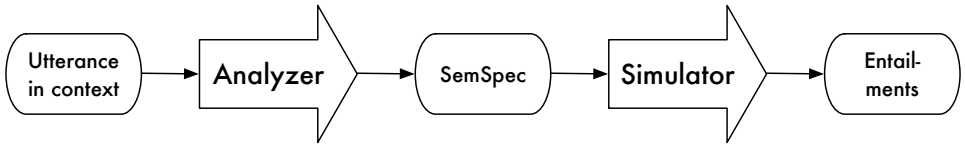

Figure 1.4: An abstract NTL language understanding architecture.

The NTL group investigates how the brain computes language. Though it is obvious that the brain uses neurons (which are a connectionist system) for processing language, NTL claims that connectionist models are the wrong level of abstraction for studying how the brain processes language. At the other end of the complexity spectrum, complex language processing systems (such as this dissertation) can only be considered cognitively plausible if the algorithms and data structures employed by the system are reducible to connectionist networks. For example, the feature structure is a computational data structure that has a connectionist implementation [19]. Its connectionist implementation is biologically plausible as well, so a feature structure is a cognitively plausible data structure.

For the constructional analyzer to be considered a cognitively plausible model within NTL, it must be reducible to connectionist models. But before one can understand the reduction of the analyzer to connectionist models, one must understand the algorithms, assumptions, and data structures employed by the system. Thus the details of the reduction will be provided at the end of the dissertation in the conclusions and future work chapter.

Within NTL, the process of language understanding is split into two functionally distinct but mutually dependent components: the analysis process and the simulation process. This dissertation defines a computational model of an analysis process that outputs a semspec. The semspec is the input to an active simulation-based model of inference [54]. Figure 1.4 shows the NTL language understanding pipeline starting with the utterance in context and ending with the results of simulation.

1.4 A Guide to the Rest of the Dissertation

The rest of the dissertation is organized as follows:

- Chapter 2 is a computational introduction to this dissertation that does not depend on a cognitive science/construction grammar background. It describes the best-fit constructional analyzer as a system that operates on feature structures, and provides an intuitive (but precise) introduction to the best-fit metric and incremental processing constraints.

- Chapter 3 covers a formal language for construction grammar called *Embodied Construction Grammar* (ECG). ECG has knowledge representation tools for precisely describing constructions, frames, and the embodied schemas. The translation between a construction in ECG and its corresponding feature structure is provided.

- Chapter 4 motivates and defines the factored probabilistic model over interpretations (also called analyses) used by the analyzer. It provides a precise definition for the syntactic, semantic and contextual models and how they are combined to score and analysis.

- Chapter 5 defines a probabilistic formulation of left corner parsing, and extends the formulation to use the factored model from chapter four. Additionally, a straightforward method for improving the speed of the search is described.

- Chapter 6 uses the analyzer as a tool for linguistic innovation within ECG. A deep semantic grammar is described, along with how it is extended for interesting "syntactic" phenomena. The semspecs for the test sentences are also shown.

- Chapter 7 shows how the analyzer can be used to make reading time predictions comparable to experimental data. The system makes reading time predictions matching the trends found by McRae, Spivey and Tannenhaus [51].

- Chapter 8 shows that the system is compatible with other languages. A hand-built grammar is used to analyze a corpus of Mandarin child-parent interactions. The Mandarin corpus is much larger than previous test cases, and this chapter shows how the best-fit metric can be used to improve the speed and accuracy of interpretation. Additionally, Mandarin is a language in which phrases that would be required in English are often omitted. This chapter shows that the analyzer can still interpret utterances that omit their arguments.
- Chapter 9 concludes the dissertation and provides directions for future research.

Chapter 2

Introduction for Computer Scientists

This chapter introduces the dissertation to computer scientists who are not familiar with cognitive science and construction grammar. It assumes that the reader is familiar with features structures and unification, basic probability theory, basic linguistic phrase types, parsing with context free grammars (CFGs), and probabilistic context free grammars (PCFGs). For an accessible introduction to these topics, please see Jurafsky and Martin [35].

Best-fit constructional analysis can be defined just in terms of operations on feature structures, and thus any unification-based theory of grammar that encodes semantic information in its grammar rules would be compatible with the parsing algorithm and feature structure evaluation metric defined in this dissertation. This chapter provides the intuition behind the feature structure evaluation metric and sketches out how the metric can be applied to incremental interpretation.

This chapter is structured as follows: Section 2.1 defines best-fit constructional analysis as a process that takes a sentence and returns a feature structure. It also describes the relevant computational properties of feature structures. Section 2.2 defines a probability model over complete feature structures, showing how to probabilistically decompose the feature structure into syntactic and semantic information. The final section sketches out how the probability model over feature structures can

be used incrementally.

2.1 From Sentences to Feature Structures

The analyzer is a system that takes a sentence as input and outputs the most likely feature structure for the input sentence given a unification grammar. The probability of a feature structure is computed in terms of the grammatical and semantic information stored within the feature structure.

To make this discussion more concrete, consider the sentence, *he slid into the room*. Figure 2.1 shows a feature structure encoding (a subset of) the grammatical and semantic information conveyed by the example sentence. The feature structure in figure 2.1 describes the example sentence as a combination of a subject subj and a finite verb phrase finiteVP. In this case, the subj role is filled by a simpler feature structure of type He. The He structure specifies its orthography with the orth role, whether it's plural or singular with the num role[1], and has a special role m which denotes the meaning of "he". In this case, the m role is filled by a function call Ref(Masc, Sing, Animate, Given) which determines the best referent for a noun phrase given the information in parentheses.[2] As is shown in the figure, the He substructure claims that its referent will be masculine, singular, animate and given in context. In this chapter, the "Ref" function encapsulates a feature structure's dependence on context. Chapter 3 and 4 provide the grammatical and computational definitions of the "Ref" function.

One basic assumption I make about feature structures is that the local semantic content of a feature structure is defined by a role m.[3] As will be shown later in this chapter, the m roles separate the syntactic content of the feature structure from the semantic content of the feature structure. This separation becomes important when

[1]For sake of simplicity, number is the only agreement feature I am including in this discussion.

[2]Including function calls as fillers of feature structure roles is not standard in unification grammar. However, the Ref call provides the intuition that a context module must be consulted before determining the meaning of *he*.

[3]In this chapter, feature structures with complex semantics like Into will have multiple m roles defined. Using multiple m roles is just a convenience in this chapter, and in general, it is not necessary as is shown in chapter 3.

Comparing unification grammar with context free grammar

	CFG	UG
Nonlocal dependencies?	No	Yes
Underlying data structure	tree	DAG
Grammar rules are	subtrees	constraint sets
Substructure Integration	symbol equality	unification
Generative Prob. Model	PCFG	?

Table 2.1: Comparing context free grammars (CFGs) and unification grammars (UGs).

I define a probability function over feature structures.

Both the num role and the m role of He have co-indexation constraints represented in boxed number notation. All roles that are marked with the same boxed number share an equality constraint on their filler. For example, the num role of the He structure, the num role of the SelfMotion structure, and the num role of the Slid structure are all co-indexed together and forced to share the "Sing" filler. Intuitively this forces agreement between the roles, and therefore allows a grammar writer to say that the subject of a sentence and the main verb of the sentence agree in number.[4]

Co-indexation ensures consistency between the substructures that make up a feature structure by introducing non-local dependencies. It is the existence of non-local dependencies (often called re-entrancy) that make unification grammar more expressive and more expensive to parse [86] than context free grammar. Table 2.1 compares context free grammars with unification grammars.

The non-local dependencies also make it difficult to define a proper generative probabilistic model over feature structures [4] [1], and why researchers began to use conditional (and in particular maximum-entropy) models for computing the likelihood of a feature structure given a sentence [33] [67] [66]. It is important to note that in all

[4]and person.

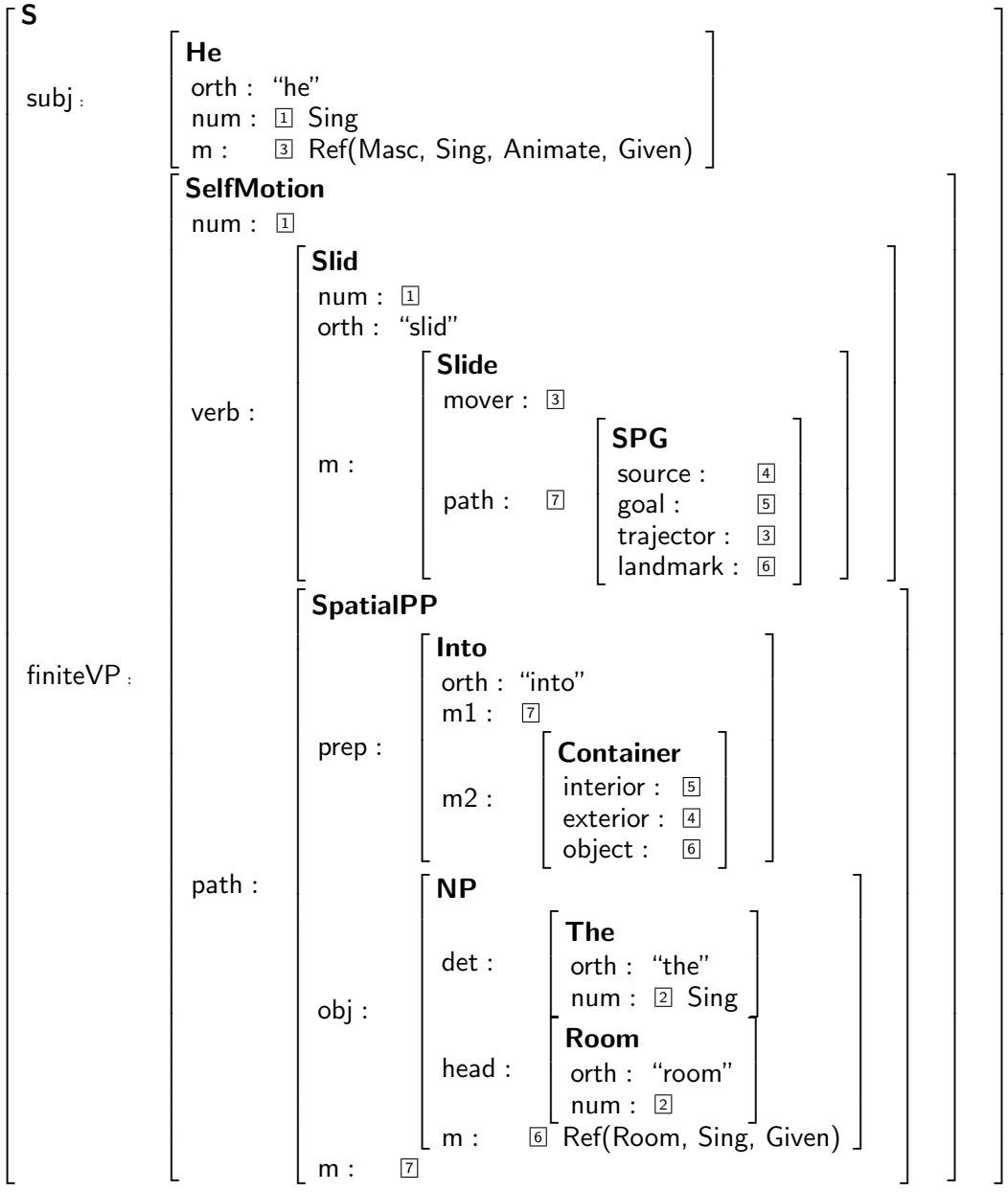

Figure 2.1: A feature structure for *he slid into the room.*

of these approaches with conditional models, the unification parser runs to completion, and then the conditional model is used to select amongst the returned feature structures. In other words, the probability model is not used online, but instead acts as a model of disambiguation. In this dissertation, a disambiguation approach cannot be used because the interpretation must happen incrementally. Additionally, as will be shown in chapter 8, improving the probabilistic model over feature structures improves the efficiency of the parser as well.

2.2 A Probability Model Over Feature Structures

The goal of the constructional analyzer is to find the most likely feature structure a given a sentence s and a grammar G or $argmax_a \Pr(\, a \mid s, G\,)$. However, I instead define a joint model[5] over the sentence and its feature structure $\Pr(\, a, s \mid G\,)$ because joint models tend to be easier to define, take the inherent structure of the problem into account (i.e. conditional independence) , and allow for simpler parameter estimation methods.

The first step in building the joint model over feature structures is to visualize them as directed acyclic graphs (DAGs). Figure 2.2 shows a DAG for the example sentence, *he slid into the room* that is equivalent to the feature structure shown in figure 2.1. In a DAG representation of a feature structure, nodes with outgoing arcs are feature structures, the (labeled) arcs are the roles, and the nodes with no outgoing arcs are atomic values such as "Sing". Co-indexation is represented by multiple arcs pointing to the same node. For example, the num roles for the NP, SelfMotion and the verb Slid are all co-indexed, and therefore all point at the same "Sing" value.

The dashed line in figure 2.2 partitions the DAG into syntactic structures and semantic structures. The only links that cross the partition are the m features, and they go from the syntactic structures to the semantic structures. This partition suggests that the semantic information can be thought of as dependent on the syntactic

[5]Collins [11] shows for the PCFG parsing that both a joint model and a conditional model define the same ranking over parses, and that the probability estimates differ only by a constant (the probability of the sentence.)

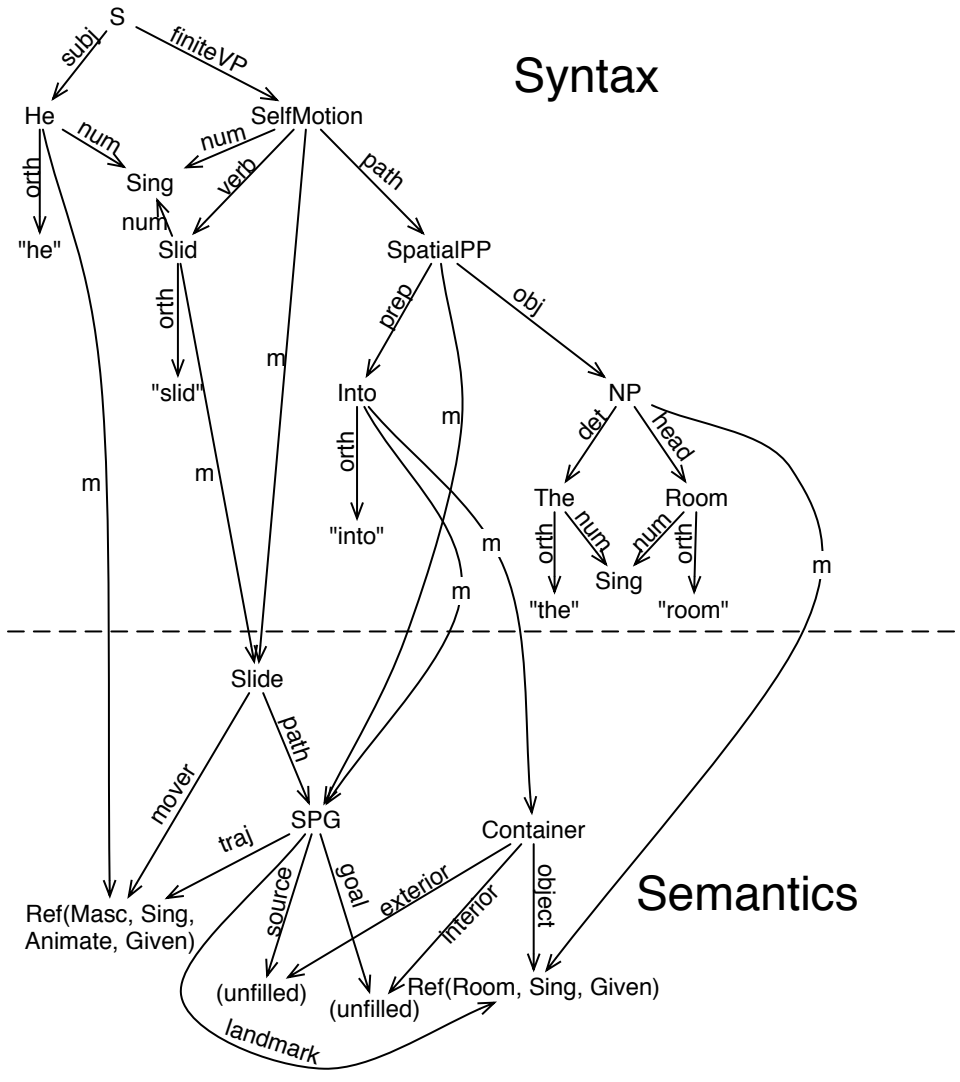

Figure 2.2: The DAG version of the feature structure shown in figure 2.1

information. Assuming that we label the syntactic information τ and the semantic structure σ, we can say:

$$\begin{aligned} \Pr(\, a, s \mid G\,) &= \Pr(\, a \mid G\,) && (2.1)\\ &= \Pr(\, \sigma, \tau \mid G\,) && (2.2)\\ &= \Pr(\, \sigma \mid \tau, G\,) \Pr(\, \tau \mid G\,) && (2.3) \end{aligned}$$

Equation 2.1 follows from the fact that the feature structure a generates the words of the sentence s. Equation 2.2 follows from the fact that σ and τ form a complete partition of the structure in a,[6] and equation 2.3 is a simple application of the chain rule of probability. Throughout this dissertation, I will refer to $\Pr(\, \tau \mid G\,)$ as the syntax factor or model and $\Pr(\, \sigma \mid \tau, G\,)$ as the semantic factor.

Figure 2.3 shows the syntactic partition τ for the feature structure associated with *he slid into the room*, and figure 2.4 shows the semantic partition σ (without the m links). Obviously τ and σ are still DAGs, so the partitioning itself does nothing to make inference or parameter estimation easier. All the partitioning really does is motivate the choice of using different independence assumptions to approximate $\Pr(\, \tau \mid G\,)$ and $\Pr(\, \sigma \mid \tau, G\,)$.

Upon closer inspection of figure 2.3, one notices that the DAG representing the syntactic information associated with *he slid into the room* looks exactly like a context-free tree if you ignore the extra agreement constraints. So the approach I use to approximate the probability of the syntactic partition τ is treat it as a context-free tree t:

$$\Pr(\, \tau \mid G\,) \approx \Pr(\, t \mid G\,) \qquad (2.4)$$

where t is the context-free version of τ. This is an independence assumption in that I am claiming that the probability of a node n in τ only depends on n's syntactic parent and not any other syntactic nodes that n might be co-indexed with. Figure 2.5 illustrates what this assumption looks like for the τ shown in figure 2.3.

Probabilistic models of context free trees are well-studied, and approximating the probability of τ using the probability of t allows me leverage work from the

[6] The definition of σ must include the m links for this to be true.

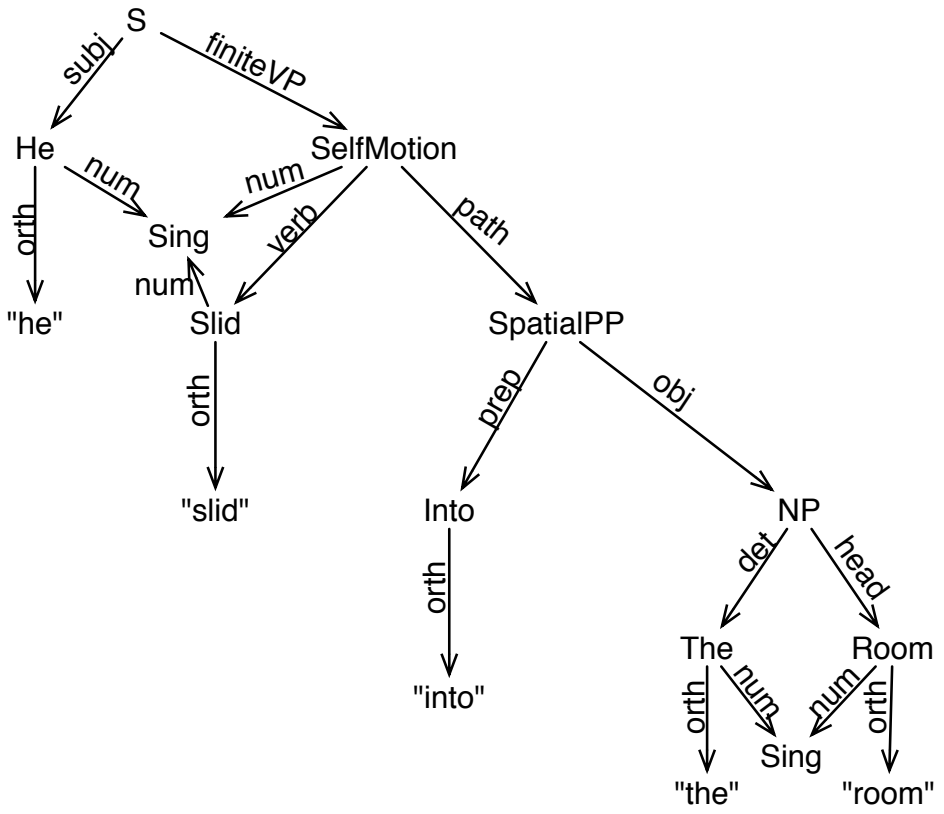

Figure 2.3: The syntactic partition of the feature structure for *he slid into the room.*

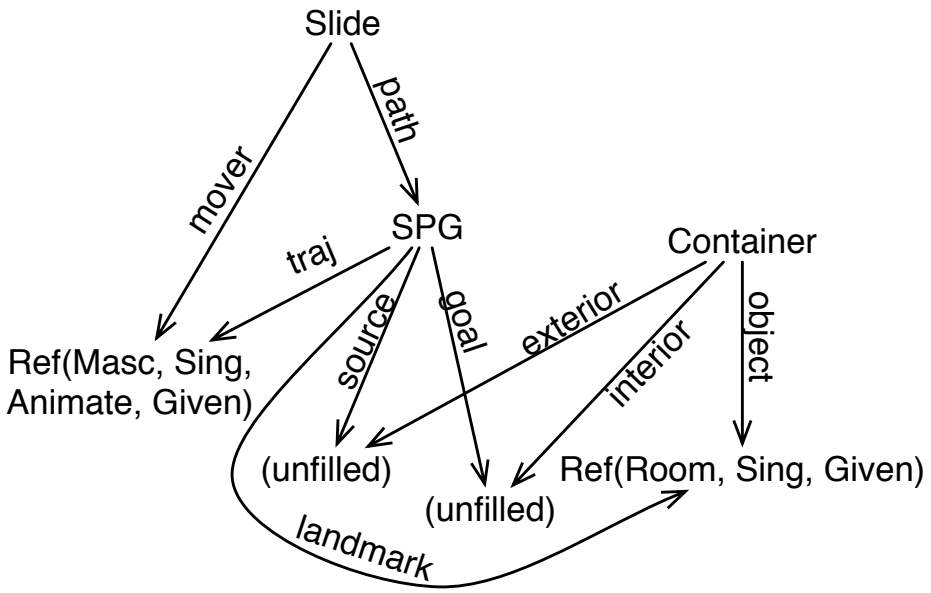

Figure 2.4: The semantic partition of the feature structure for *he slid into the room.*

statistical parsing community. Collins' work on generative models of syntactic parsing [11] is particularly influential, and in chapter 4, I define a probability model over constructional trees t that uses his work as a starting point.

For the semantic partition σ, the nonlocal constraints are the important ones, and thus a completely different approximation strategy is employed. Looking again at the complete DAG a shown in figure 2.2, one can see that the syntactic partition τ specifies much of the structure of σ through the m arcs. For example, the Into rule deterministically generates SPG and Container structures if it is used in τ. Similarly, the Slid rule deterministically generates the Slide semantic structure.

In fact, given a complete τ, all of the structure of σ is deterministically specified, but each of the rules that went into building τ has a myopic perspective, only knowing about its local unification constraints. Consider the example S rule shown in figure 2.6. It deterministically links the meaning of its subj role to the mover role of its

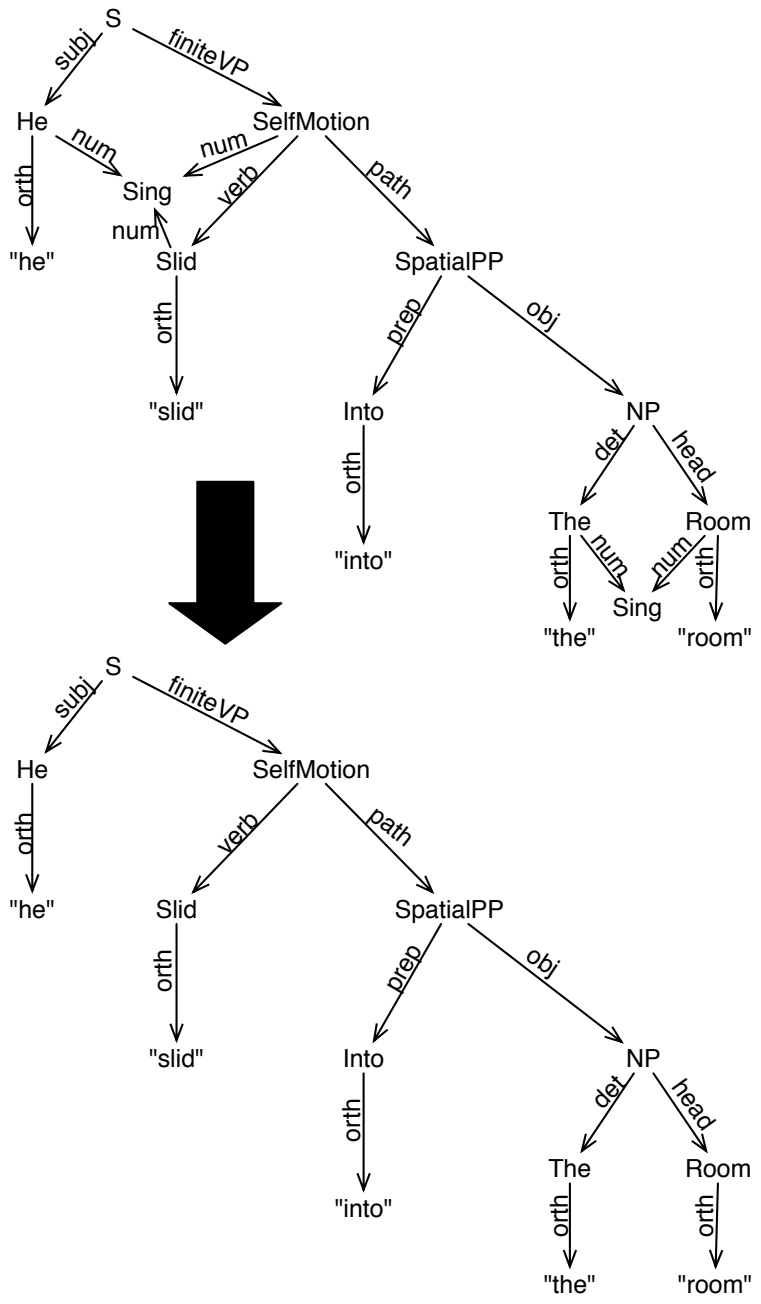

Figure 2.5: An illustration of the independence assumptions made by the syntax model.

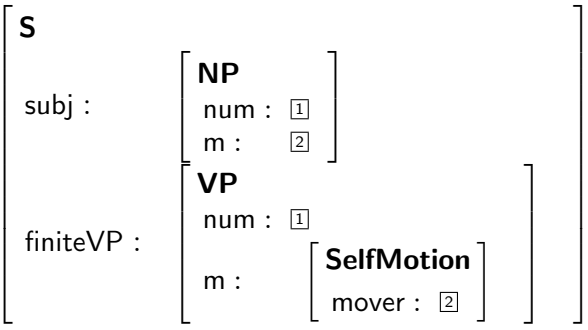

Figure 2.6: An example SelfMotion-specific S rule that binds the meaning of its subject constituent to the mover of its finiteVP constituent.

finiteVP.[7] However, the fact that the constraint results in the mover role being filled by Ref(Masc, Sing, Animate, Given) is not local to any particular rule in the grammar. Nonlocal constraints like the mover role being filled by Ref(Masc, Sing, Animate, Given) have not been taken into account when computing the probability of τ, and thus the nonlocal semantic constraints are what determine $\Pr(\sigma \mid \tau, g)$.

The insight that the nonlocal semantic fillers determine the likelihood of σ will only take us so far without further independence assumptions. Going back to the semantic partition shown in figure 2.4, we can see that the probability of σ shown in figure 2.4 given the τ shown in figure 2.3 can be defined as the joint distribution over all the role=filler pairs:

$$\begin{aligned}\Pr(\quad & Slide.mover = Ref(Masc...), \\ & SPG.traj = Ref(Masc...), \\ & SPG.landmark = Ref(Room...), \\ & Container.object = Ref(Room...), \\ & Slide.path = SPG \qquad\qquad \mid \tau)\end{aligned}$$

There are various independence assumptions that we could make to approximate $\Pr(\sigma \mid \tau)$. One such assumption would be to treat each of the role-filler pairs as

[7] Obviously in a real grammar, one would not have different S rule for every unique VP rule. A more general S rule will be defined in chapter 6, and it binds the meaning of its subject to a generalization over all semantic roles called profiledParticipant.

independent. However, that would heavily penalize a more complete semantic structure which would conflict with the idea that adding more semantics to grammar is a good thing. Instead of treating each role=filler pairing as independent, I just treat the different fillers as independent. For our example σ, this amounts to:

$$\begin{aligned}\Pr(\sigma \mid \tau) \approx\ & \Pr(Slide.mover = Ref(Masc...), SPG.traj = Ref(Masc...) \mid \tau) \times \\ & \Pr(SPG.lm = Ref(Room...), Container.obj = Ref(Room...) \mid \tau) \times \\ & \Pr(Slide.path = SPG \mid \tau) \end{aligned} \tag{2.5}$$

And then I further simplify equation 2.5 by computing the probability of a binding with a potential function F. Function F assumes that each of the bindings are independent of each other and τ. For our example σ, $\Pr(\sigma \mid \tau) \approx$

$$\begin{aligned}& F(\Pr(Slide.mover = Ref(Masc...)), \Pr(SPG.traj = Ref(Masc...))) \times \\ & F(\Pr(SPG.lm = Ref(Room...)), \Pr(Container.obj = Ref(Room...))) \times \\ & F(\Pr(Slide.path = SPG)) \end{aligned} \tag{2.6}$$

Figure 2.7 illustrates what the semantic partition looks like after undergoing all these independence assumptions. Section 4.4.4 provides further motivation for this decomposition as well as defining the parameters that go into calculating the probability of a role=filler pairing.

2.3 Incremental Processing

So far I have shown how to compute the probability of a complete feature structure a. In chapter 1, however, I promised an incremental model of interpretation, and thus the assumption that feature structure a is complete does not hold during the process of interpretation itself. The process of incremental interpretation (covered thoroughly in chapter 5), processes each word of an utterance in order from the beginning of the utterance to its end, and as each word comes in, its syntactic and semantic constraints are incorporated into feature structure a and the probability estimate of a must be updated.

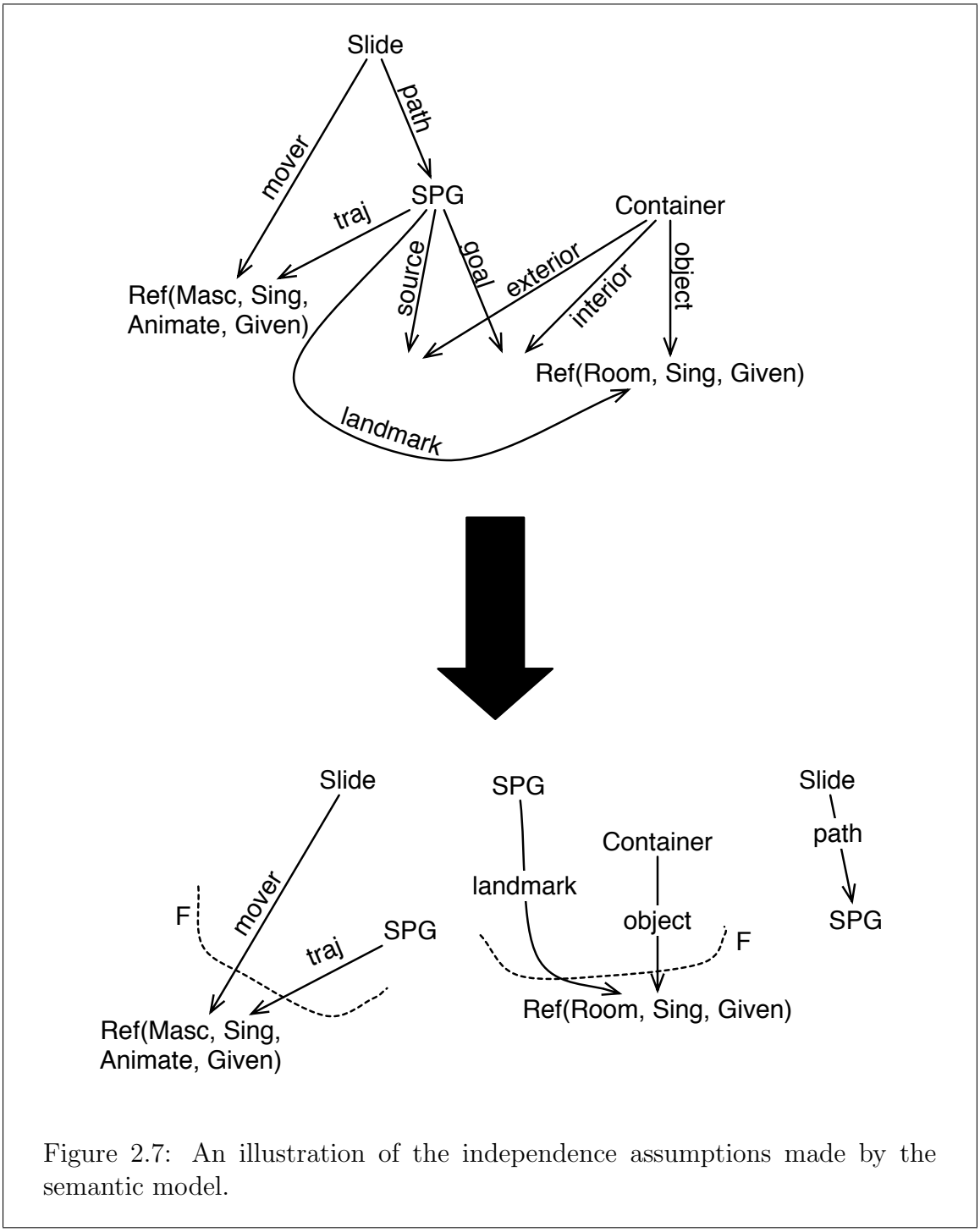

Figure 2.7: An illustration of the independence assumptions made by the semantic model.

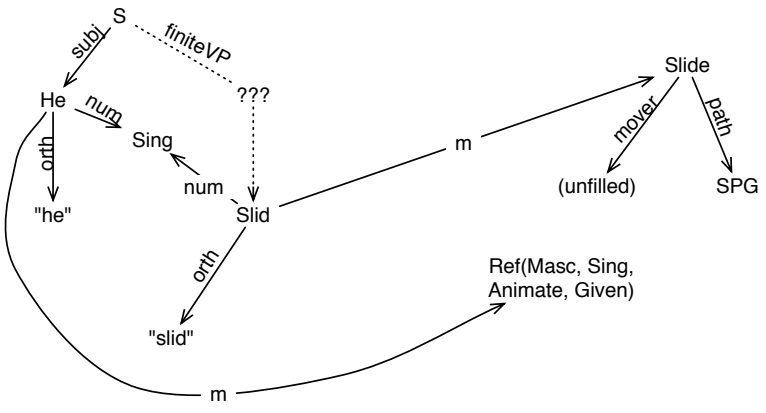

Figure 2.8: For the prefix, *he slid*, the system is not sure if the meaning of He is bound to the mover role.

The problem with updating the probability of a after each word is that the syntactic partition τ will be incomplete until the end of the sentence, and uncertainty in τ leads to structural uncertainty in σ. And if there is structural uncertainty in σ, then there is uncertainty about which roles are bound to which fillers and the semantic factor as defined so far cannot evaluate σ.

To make this discussion more concrete, consider figure 2.8 which shows the partial DAG for the prefix *he slid*. The figure illustrates the uncertainty about the kind of VP construction that will connect the S node with the Slid.[8] Uncertainty in the choice of VP construction leads to uncertainty about the semantic structure. If the next word is *into*, as it is in the original example sentence, Ref(Masc...) will be bound to the mover role. But if the next word is *the*, as in *he slid the puck across the ice*, then Ref(Masc...) will instead be bound to the causer role as is shown in figure 2.9.

But the system can still make progress with all this uncertainty. The system first computes the probability of the partial τ using the incremental probabilistic left-corner parsing algorithm [48]. The probabilistic left-corner parsing algorithm computes the probability of the context-free approximation of τ (tree t) in terms of

[8]Technically, the system should also be unsure about the co-indexation of Slid's num role, but if every way of connecting S and Slid co-indexes the two num roles, then the system will know that they are always unified.

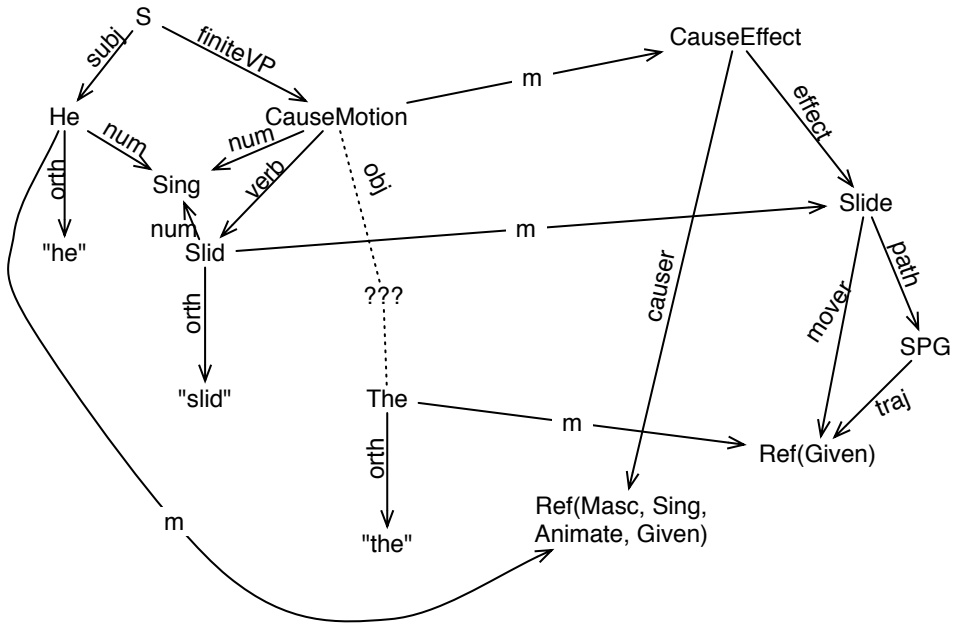

Figure 2.9: For the prefix, *he slid the*, the meaning of He is bound to the causer role.

the probability of each parser operation used to build t. Chapter 5 describes the left corner parsing algorithm and the probabilistic variant I use in the system.

With an incremental probabilistic parser, the system can compute the probability of a partial τ. What remains, then, is to estimate $\Pr(\sigma \mid \tau)$ in the incremental setting when τ (and therefore σ) is only partially complete. The key idea here is that while τ is only a partial specification of the syntactic information, it still carries enough information to compute the likelihood of various completions of σ called σ_c. Because each σ_c is a complete σ, we can use the technique from the last section to estimate its (inner) likelihood.

This process can be illustrated by returning to the *he slid* example shown in figure 2.8. Assume that the probabilistic left-corner parsing algorithm has returned the probability of the syntactic partition τ' in figure 2.8. Figure 2.10 shows the two possible completions of σ' given τ'. The top completion corresponds to the finiteVP being filled by SelfMotion and the bottom completion corresponds to the finiteVP being filled by CauseMotion.[9]

Assuming that SelfMotion and CauseMotion are the only two possible VPs given τ', the probability of the SelfMotion interpretation is:

$$\Pr(Slide.mover = Ref(Masc...), SPG.traj = Ref(Masc...) \mid \tau', SelfMotion) \times \Pr(SelfMotion \mid \tau') \Pr(\tau' \mid G) \quad (2.7)$$

And the probability of the CauseMotion interpretation is:

$$\Pr(NoBindings \mid \tau', CauseMotion) \Pr(CauseMotion \mid \tau') \Pr(\tau' \mid G) \quad (2.8)$$

Equation 2.7 computes the likelihood of the Slide.mover=Ref(Masc...) interpretation as a product of the probability of τ', the probability of the SelfMotion construction given τ', and the probability of the complete resulting σ that assigns the meaning of He to the mover and traj roles. Equation 2.8 multiplies the probability of τ' by the probability of the CauseMotion construction given τ' by the probability of no bindings in the completed σ.

[9] As in the *slid the puck* example.

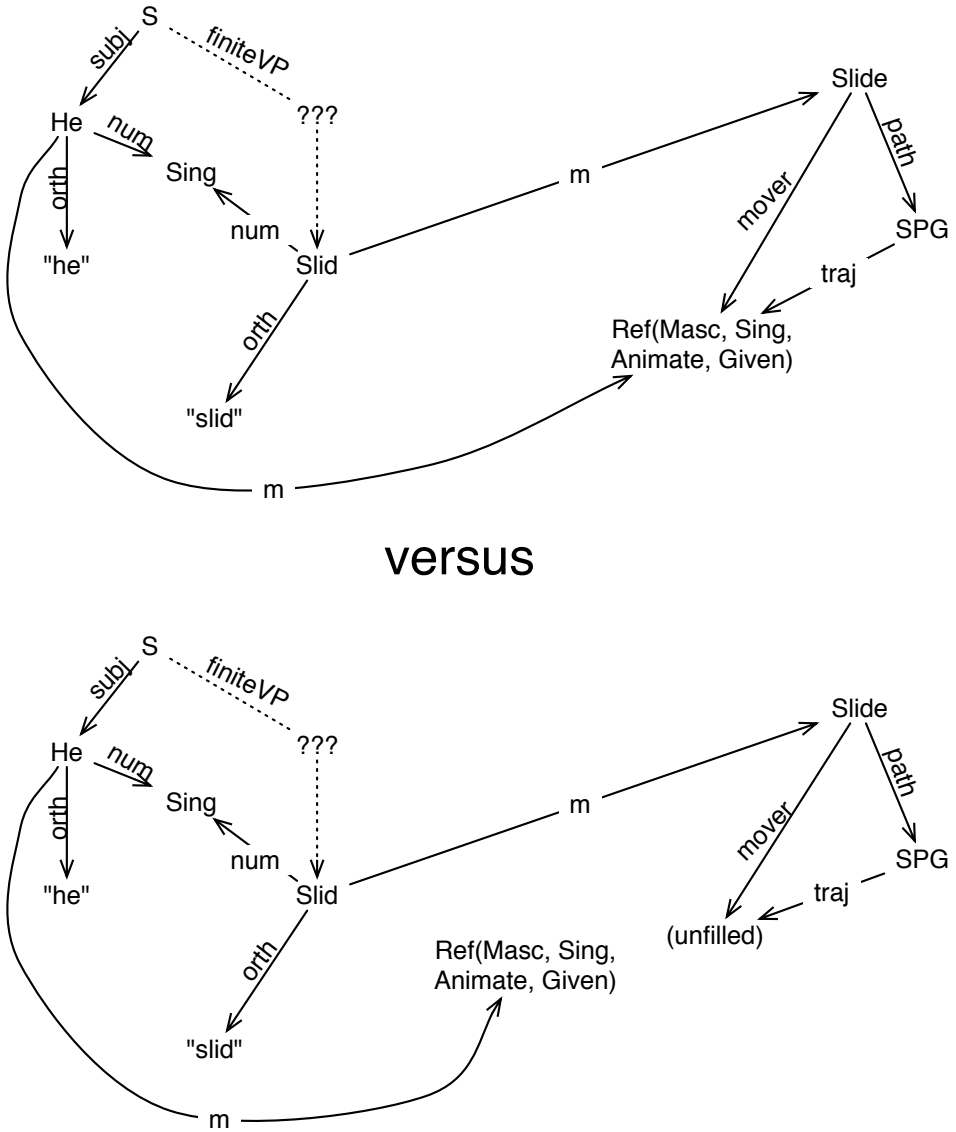

Figure 2.10: For the prefix, *he slid*, there are two possible semantic completions.

The assumption that the only two possible ways of connecting S and Slid is obviously a simplification. In the general case, the equations above would sum over all the ways that S and Slid connect up and generate the same interpretation. Chapter 5 shows how to pre-compute the paths and interpretations when there is uncertainty about how two constructional nodes in the DAG are connected.

2.4 Summary

This chapter provides the intuition behind a factored probabilistic model over feature structures and shows how it can be used during incremental processing. The factored model partitions the syntactic and semantic information in the feature structure, and uses independence assumptions to efficiently estimate the probability of each factor. Chapter 4 provides more detail on the factored model.

One of the virtues of a factored model is how it can be made incremental, as is shown in this chapter. Because the model is incremental, it can be used online to guide the parsing process. Chapter 5 shows precisely how the factored model is made incremental by deriving a semantic extension to the left-corner parsing algorithm. Chapter 7 shows the factored model is used to predict incremental reading time difficulty. Chapter 8 shows how using machine learning to infer the parameters in the factored model improves the speed and accuracy of the system on a Mandarin corpus.

Chapter 3

Implementing Embodied Construction Grammar

This chapter provides a brief introduction to construction grammar and a thorough introduction to *Embodied Construction Grammar* (ECG) [3]. As a notation for defining construction grammars, ECG specifies constructions in terms of form constraints and meaning constraints. ECG is embodied because the meanings constraints are defined in terms of neurally motivated semantic schemas.

Importantly, ECG is a precise knowledge representation language for constructions and schemas. In addition to defining the syntax of ECG, this chapter shows how the analyzer pre-processes a construction grammar before analysis starts. First it reads it from text into a grammar data structure, checking it for consistency. If the consistency checks are passed, the analyzer turns the grammar into a set of feature structures. Each feature structure has all the same constraints as the original text specification, and is easy to adapt to the process of best-fit interpretation.

Section 3.1 discusses some of the important properties of construction grammar. Section 3.2 provides a qualitative introduction to the ECG formalism using examples. Section 3.2.2 shows how ECG has been extended since the original publication of the formalism by Bergen and Chang. Section 3.3 precisely defines the implemented version of ECG and how a construction grammar is pre-processed for use with the constructional analyzer.

3.1 Construction Grammar

Construction grammar is a theory of grammar that defines each grammatical rule (*construction*) as a mapping between a form and a function [21]. Thus grammaticality is defined not just in terms of syntactic constraints, but also in terms of semantic/pragmatic constraints.

Because grammaticality is defined in terms of both form and meaning, construction grammar is in direct opposition to modular approaches to language. In addition, there is no notion of movement within the theory of construction grammar making it a simpler syntactic theory than movement based approaches [14].

Researchers in the cognitive field have extended construction grammar to incorporate cognitive constraints as well. Goldberg, for example, defines constructions in terms of radial categories [26]. Tomasello provides a usage-based model describing how children learn language [85]. Bergen and Chang show how construction grammar can be linked to simulation semantics in a way that is both cognitive and precise [3].

Figure 3.1 shows four example constructions represented in an intuitive way. The three columns in figure 3.1 show the name of the construction, a description of its form and a description of its meaning. The Affix -ed construction is a morphological construction that links the affix *ed* with the verbal semantics associated with simple past tense. The lexical Give construction shows that a word like *give* is linked to some kind of representation of the give action.

The Double Object construction (based on Goldberg [26]) is an interesting clausal construction because it associates the double object form with both a notion of Transfer and a set of constraints that link the constituents of the construction to the various participants in a Transfer scene.[1] The double object construction shows how a construction's semantics is not purely compositional. Consider the following two sentences:

- Bob gave me a book.
- Bob baked me a cake.

[1]For example, the subject NP_0 gets linked to the *giver* role of the Transfer.

Name	Form	Meaning
Affix -ed	"-ed"	past speech time, completed action
Lexical Give	"give"	a give action
Double Object	NP_0 V NP_1 NP_2	Transfer Scene + bindings
WXDY	What is NP_0 doing PP_0	How come NP_0 is PP_0

Figure 3.1: Intuitive descriptions of four example constructions

The first sentence is a prototypical example of the double object construction. The word "gave" has three semantic arguments: a giver (Bob), a recipient (the speaker), and a gift (a book). One can say that the meaning of the whole sentence looks like a simple combination of its constituents' meanings.

Bob baked me a cake is a non-central example of the double-object construction. The notion of transfer implied by the sentence is not present in any of the words in the sentence, and the recipient is not even a semantic participant in the meaning of the word *bake*. Goldberg shows that the notion of transfer and the binding of *me* to an intended recipient role could only license by the Double Object construction, and therefore the semantics of this construction are not purely compositional [26].

Goldberg goes on to show that the double object construction that licenses *give* and the double object construction that licenses *bake* are members of a complex family of constructions ordered into a radial category [26]. Lakoff shows how radial categories can be applied to constructions in his case study of the *There* construction [43].

The WXDY construction is a shorthand for the *What is X doing Y?* construction described by Kay and Fillmore [38]. An example of this construction is the sentence *What's a nice girl like you doing in a place like this?*. As Kay and Fillmore point out, this question is not asking about the activity that the nice girl is doing, but rather has a paraphrase more like *How come a nice girl like you is in a place like this?* which suggests that the situation is inappropriate for some reason. Notice that this meaning is also non-compositional (i.e. not purely a function of the words used

in the sentence), and that this ambiguity between the compositional meaning and the non-compositional meaning is the reason for the humor in the following exchange:

- Waiter! Waiter! What's this fly doing in my soup?
- It appears to be doing the backstroke, sir.

3.2 Embodied Construction Grammar

In our discussion so far, the constructions have been represented in an informal way. A handful of construction grammar approaches address the need for a formal representation. One such example is Kay and Fillmore's Construction Grammar (CxG) which is defined in terms of typed feature structures [37], but their theory is not constrained by cognitive grounding. This is where Embodied Construction Grammar (ECG) comes in [3].

ECG is a different kind of construction grammar precisely because it cares both about cognition *and* being precise with the notation. The goal of the formalism is to be expressive enough to provide both cognitively plausible analyses of all aspects of form (from morphemes to clauses to gestures to intonation etc), while acknowledging that the grammar must be constrained by limitations of the (human) parser and vice versa.

There are two key principles that underlie the design of the ECG formalism. The first principle is that the semantics of every construction is defined in terms of cognitively plausible structures like image schemas [82], frames [20] and metaphors [44]. The second principle is that language understanding is a cognitive process called *simulation*. More precisely, the motor programs in the brain that are used to execute actions are also used to understand language. These motor programs take parameters to control their execution, and language is one way to specify those parameters. These two principles are what make ECG special. ECG isn't just some arbitrary formalism, but rather a formalism that is grounded in cognition through embodied structures and simulation.

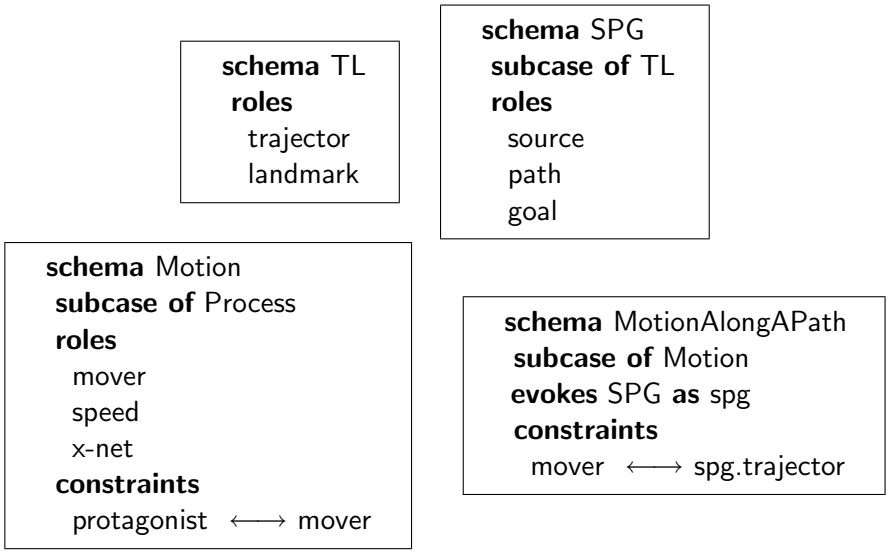

Figure 3.2: A simplified subset of the ECG schemas required to represent the meaning of the sentence *Eve walked into the house*. TrajectorLandmark and SPG are image schemas related through the *subcase of* keyword. The MotionAlongAPath schema is a kind of Motion schema which is a kind of Process.

3.2.1 ECG as a Knowledge Representation Language

Constructions are *structured knowledge* about language. ECG specifies this structured knowledge with two basic primitives.[2] *Constructions* specify constructions and *schemas* represent linguistically relevant meaning. There are four ways to specify relations between ECG structures: roles, sub-typing (through the *subcase of* keyword), evoking a structure (through the *evokes* keyword) and constraints (co-indexation and typing). Intuitively speaking, roles name parts of a structure, and sub-typing allows for (partial) inheritance.

Evoking a structure makes it locally available without imposing a part-of or subtype relation between the evoking structure and the evoked structure. The canonical example usage of *evokes* concerns the definition of the concept *hypotenuse*. The con-

[2]There are two additional primitives that are the subject of ongoing research: Maps and Situations. Situations are the ECG correlate to mental spaces, and maps are structures that control information flow between domains such as frames and situations.

cept hypotenuse only makes sense in reference to a right triangle, but a hypotenuse is not a kind of a right triangle, nor is the right triangle a part of (role) of the hypotenuse. *Evokes* is used to state the relationship between the hypotenuse and its right triangle.

ECG also has operators for co-indexing roles ($\longleftrightarrow$) as well as assigning atomic values to roles ($\leftarrow$). Type constraints specified with a colon constrain the fillers of roles, and the **self** operator lets a schema or construction refer to itself.

In figure 3.2, the TrajectorLandmark schema has roles for the trajector and the landmark. Motion schema is a Process (Process is not shown) that adds three roles and co-indexes the Process.protagonist role with the mover role. The x-net role denotes which embodied motor program to use when simulating the motion (e.g. walking, crawling, sliding etc.). The MotionAlongAPath schema uses the evokes keyword to activate an instance of the SPG schema to represent the path and then binds the mover role to the spg.trajector role, thereby denoting that the mover is the trajector moving along a path.

In figure 3.3 there is an example MotionAlongAPath construction. The MotionAlongAPath construction covers utterances like *The man walked into the house*. Constructions are arranged into an inheritance lattice much like schemas. The MotionAlongAPath construction is a subcase of the VerbPlusArguments construction which combines (both syntactically and semantically) a verb form with its constituents. The VerbPlusArguments construction is a subcase of ArgumentStructure which is the root of the VP hierarchy and is more general than a single verb plus its arguments.[3]

Every construction is defined with three (optional) blocks: a constructional block, a form block and a meaning block. The constructional block specifies constituency and constructional features and constraints (for example, specifying number agreement between subject and object). The form block places ordering constraints on the constituents, specifying a partial ordering over constituents. The meaning block specifies its denotation through its meaning type and adds any additional meaning.

Because the MotionAlongAPath construction is defined to be a kind of VerbPlusAr-

[3]A construction marked with the general keyword is not used directly in analysis, but is there just for typing and inheritance.

guments construction, it has two constituents, one inherited constituent for the verb v and one for constituent pp. Each constituent is constrained to be a particular construction type. Constituent pp, for example, is typed to be an Spatial-PP which are spatial prepositional phrases.

Form constraints employ operators that constrain the relative ordering of constituents. One such operator is the *before* operator which requires its left argument be before, but not necessarily immediately before its right argument. Another such operator is the *meets* operator which asserts that its left argument be immediately before its right argument. MotionAlongAPath's form block asserts the relative form constraints, in this case specifying that constituent v comes before constituent pp. The .f notation indicates (semi-redundantly) that the form constraints are constraining the form pole of the mentioned constituents.

The construction's meaning pole specifies that it denotes a MotionAlongAPath scene. Then within the meaning block, the meaning of each constituent is tied to the corresponding role in the scene. The meaning of constituent v is co-indexed to the self.m role of the construction. This means that the meaning of the verb (referenced using the .m notation) is bound to the meaning pole of the MotionAlongAPath construction. The keyword self indicates a reference to the containing construction, and like before, .m refers to the defined meaning type of a construction. Similarly, the meaning pole of the pp constituent is bound to the spg role because the pp role specifies the path of the motion.

3.2.2 Extended ECG

The previous section described the syntax of the ECG formalism and showed some examples of how it could be used. While the original formulation specified by Bergen and Chang [3] was sufficient for a simple child language model [9], the formalism has been extended to better handle reference and different kinds of optional elements. These ECG extensions also lead to more intensive processing requirements beyond those implemented in the previous version of the system [5].

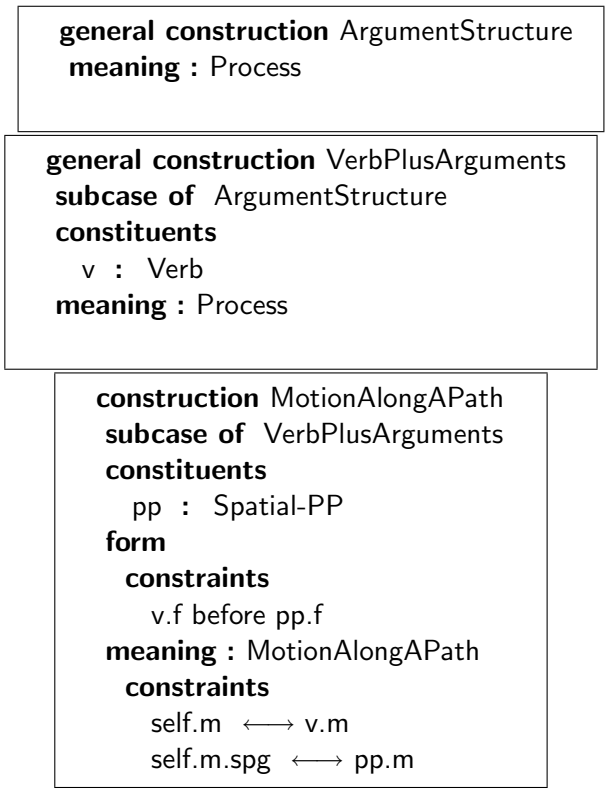

Figure 3.3: A MotionAlongAPath construction. This construction describes active voice VPs with a motion verb and a PP path specifier.

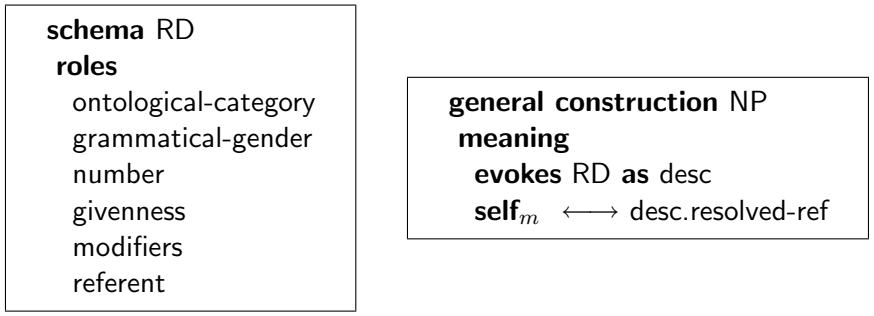

Figure 3.4: An RD schema and an abstract NP construction that only specifies meaning constraints.

Reference

To ground an analysis in terms of the "real world", the analyzer must also interact with a context model that implements a reference resolution engine. To resolve a reference, the context model takes as input a set of semantic and syntactic constraints associated with a reference, and returns some referent to unify into the analysis. To aid this process, a *ReferentDescriptor* schema or RD is defined. The RD captures the constraints associated with a particular reference, and thus functions as the interface between the analysis and the reference resolution.

Figure 3.4 shows the RD as well as a general NP construction. The RD has roles for the ontological category, grammatical gender, modifiers–basically anything that is used to tie a reference to its referent. Another important role in the RD is the referent slot which is filled by the referent proposed by the context model. The abstract NP construction shows how to use the RD and the referent slot. It evokes the RD in its meaning pole, and then binds its meaning pole to the referent slot. After resolution, the meaning of the NP is the resolved referent. Such an arrangement makes the RD transparent with respect to the frames that link up to the meaning of the NP.

```
construction MotionAlongAPath
 subcase of VerbPlusArguments
 constituents
   v : Verb
   pp : Spatial-PP
  optional temp : TimeAdjunct
  optional loc : LocationAdjunct
 form
  // same as before
 meaning : MotionAlongAPath
  // same as before
```

Figure 3.5: A MotionAlongAPath construction with two optional constituents. Constituent temp specifies an optional constituent for a time adjunct while loc specifies an optional constituent for a locative adjunct.

Optional Elements

There are two basic notions of optionality. The first is when a construction *expects* a constituent to be present, but it isn't there. Gapping and omission of complements are the linguistic phenomena associated with this kind of optionality. The assumption made here is that all of constituents of a construction that are not marked as optional are complements, and that any of these complements could possibly be gapped/omitted, though with differing frequency. The structural aspect of the grammar does not need to be extended to handle omission/gapping. However, this approach does require that frequency information describing omission and gapping preferences be specified along with the grammar.

The second kind of optionality is when a constituent of a construction is always completely optional. This is often the case with adjuncts and adjectives. To handle adjuncts, ECG has the keyword optional. Optional constituents can be added to the constituents block, participate in form constraints and be semantically bound just like any other constituent. Their primary difference is that they don't have to be recognized before the construction can be considered complete. An example of the optional keyword is shown in figure 3.5.

```
construction Fronted-Wh-Question
 constructional
  constituents
   extraposed qnp : WH-NP
    fin : Aux
    subj : NP
    argstruct : ArgumentStructure
 form
  constraints
    qnp.f meets fin.f
    fin.f meets subj.f
    subj.f meets argstruct.f
 // Meaning block to be defined in chapter 6
```

Figure 3.6: A example construction that uses the extraposed keyword. This example Fronted-Wh-Question has four constituents. A WH-NP called qnp covering phrases like *which book*. A finite auxiliary constituent named fin. A subject NP called subj, and an ArgumentStructure construction called argstruct. The qnp constituent is marked by the extraposed which denotes that it is available as an argument for whatever construction fills the argstruct role.

Nonlocal Specification of Constituents

Some constructions of English express a semantic argument of another construction in an analysis. For example, in the sentence *Which ticket did you buy?*, the meaning of the phrase *which ticket* is the semantic argument of the buying event. In this case, the question construction takes a WH-NP constituent, but the meaning of that constituent has to trickle down from the question construction into the VP headed by *buy*. This allows the VP headed by *buy* to not match its object constituent and semantically bind its goods role to the meaning of *which ticket*.

A special ECG keyword extraposed is available for specifying which constituents have the ability to trickle down and act as an constituent of another construction. A construction with a constituent that is fronted with respect to some other construction marks that constituent with the extraposed keyword. Figure 3.6 discusses an example wh-question construction that employs the extraposed keyword.

3.3 The Implemented Syntax of ECG

The qualitative description of ECG was suggestive of what a well-formed ECG grammar might look like, and in this section I provide an ECG well-formedness specification. Because ECG is a formal language, I employ a context free specification of its syntax. Grammars that satisfy this context free specification shown in figure 3.7 are not necessarily well-formed, but given a grammar that does specify the context free specification in figure 3.7, it is straightforward to perform further checks.

In the context free specification in figure 3.7, fully capitalized terms are the lexical items of the grammar, while the nonterminals are in lower case. The | operator separates productions and ϵ is a production that goes to the empty string. The production for Schema is split onto the next line. Nonterminals with a capitalized "L" in them are lists and nonterminals with "Opt" are optional because they have an ϵ production. Additionally, ECG keywords such as "schema" and "form" are case insensitive. In other words, "schema", "SCHEMA" and "sChEmA" all match the terminal SCHEMA.

A closer look at figure 3.7 reveals my assumptions about structurally well-formed ECG grammars. Some of these assumptions are no different than the official ECG specification. For example, I assume that a construction (Cxn) is comprised of a the keyword "construction", followed by an identifier specifying the construction's name, followed by an optional parent list (ParentLOpt), followed by an optional constructional block (CxnBlockOpt), an optional form block (FormBlockOpt), and an optional meaning block (MeaningBlockOpt).

More interesting, however, are the specifications for the different kinds of blocks. While the meaning block looks just as expected, the form and the constructional blocks have additional properties.

The constructional block can be typed to make constructional features such as person and number available.[4] Following the constructional block type, the constituents are defined. And finally, optional constructional constraints can specified over the constructional features. Importantly, evoked constructions are not allowed

[4]It is not allowed, however, to define constructional features in the constructional block directly.

ECGL	→	ECGL Schema | ECGL Cxn | ϵ
CxnKind	→	GENERAL CONSTRUCTION | CONSTRUCTION
Cxn	→	CxnKind IDENT ParentLOpt CBlockOpt FBlockOpt MBlockOpt
BlockType	→	: Typespec | ϵ
CBlockOpt	→	CONSTRUCTIONAL BlockType ConstitsLOpt ConstraintLOpt | ϵ
FBlockOpt	→	FORM BlockType ConstraintLOpt | ϵ
MBlockOpt	→	MEANING BlockType EvokedLOpt RolesLOpt ConstraintLOpt | ϵ
SchemaKinds	→	FEATURE SCHEMA | SEMANTIC SCHEMA | SCHEMA
Schema	→	SchemaKinds IDENT ParentLOpt EvokedLOpt ...
		RolesLOpt ConstraintLOpt
ParentLOpt	→	ParentL | ϵ
SubcaseOf	→	SUBCASE | SUBCASE OF
ParentL	→	ParentL , IDENT | SubcaseOf IDENT
Typespec	→	IDENT | EXTERNALTYPE
EvokedElement	→	EVOKES Typespec AS IDENT
EvokedLOpt	→	EvokedLOpt EvokedElement | ϵ
Role	→	IDENT OptType
OptType	→	: Typespec | ϵ
RolesLOpt	→	RolesL | ϵ
RolesL	→	RolesL Role | ROLES
Constit	→	OPTIONAL IDENT : IDENT OptConstitAnno
	|	EXTRAPOSED IDENT : IDENT OptConstitAnno
	|	IDENT : IDENT OptConstitAnno
OptConstitAnno	→	[ProbL] | ϵ
ProbL	→	PROB | PROB , PROB
ConstitsLOpt	→	ConstitsL | ϵ
ConstitsL	→	ConstitsL Constit | CONSTITUENTS
ConstraintLOpt	→	ConstraintL | ϵ
ConstraintL	→	ConstraintL OptIgnore Constraint | CONSTRAINTS
ChainOperator	→	$\longleftrightarrow$ | BEFORE | MEETS
OptIgnore	→	IGNORE | ϵ
Var	→	SLOTCHAIN | IDENT
Constraint	→	Var ChainOperator Var | Var $\longleftarrow$ IdentOrStr
IdentOrStr	→	EXTERNALTYPE | IDENT | STR
IDENT	→	[A-Za-z][0-9a-zA-Z-_]*
SLOTCHAIN	→	(IDENT.)+IDENT
EXTERNALTYPE	→	@[0-9a-zA-Z.-_]+
STR	→	“(\” | [∧\n”] | \{WHITE_SPACE_CHAR}+\)*”
PROB	→	.[0-9]+ | 1.0 | 1

Figure 3.7: A context free representation of the accepted ECG syntax.

in the constructional block.

Form blocks are the least structured. A form block can be typed to make features available. This is how the "orth" feature is defined for lexical items. Additional form features can be defined to specify phonological properties as well, but these are not used in the system. While ECG allows eight different form constraints, the only two constraints currently supported in the analyzer are before and meets. Assignment and co-indexation constraints are also allowed in the constraints block to bind values to form features.

Defining Constituents

Constituent definitions are also extended in the implemented version of ECG in that there are now three kinds of constituents:

- Regular (unmarked) constituents are the core constituents of a construction. The analyzer assumes that unmarked constituents must either be specified in the input or defined in context, and thus if they are omitted, reference resolution is used to find the missing semantic component. Regular constituents are allowed to be expressed nonlocally (e.g. fronted), though the likelihood of this for most constituents is zero.

- Optional constituents, marked with the keyword optional, are more like adjuncts in that they can be omitted without any additional reference operations being instantiated. Optional constituents cannot be expressed nonlocally. Temporal modifiers or adjectives might be kinds of optional constituents. And like any other kind of constituent, optional constituents are inherited by subtypes.

- Constituents marked by the keyword extraposed, are special constituents that (within an analysis) are syntactically co-referent with a nonlocal constituent of some other construction in the analysis. In other words, the constituent marked with *extraposed* is extraposed with respect to some other construct. Constituents marked as *extraposed* cannot be omitted and must be expressed

locally in the construction that defines them. For example, the Fronted-Wh-Question construction with a subj-aux inversion (shown in figure 3.6) that covers the sentence *Which ticket did you buy?* has an extraposed constituent that would correspond to the object of the VP headed by *buy*. In other words, if a construction with a constituent marked with *extraposed* is used in an analysis, then the syntactic context associated with fronting is set up.

By default each constituent is set to be omitted with probability zero and specified locally 100% of the time. One can modify this using the optional constituent annotation (OptConstitAnno) by specifying either one or two probabilities in brackets. The first probability is the likelihood that the constituent is expressed (i.e. 1 - the probability of omission) and the second probability is chance that the constituent is specified locally (i.e. 1 - the probability of it being fronted).

The Special Assignment Constraint

The implemented version of ECG has special functionality associated with the ⟵ (assignment) constraint. In addition to being used to assign an atomic value to a role (atomic values are specified as quoted strings), the grammar writer can use the single-headed arrow to assign anonymous instances. To do this, the grammar writer specifies a type as the right hand side argument of the assignment, and the analyzer assumes that the role is being filled by an anonymous instance of the type. This makes it possible for the grammar writer to assign a type to a role without having to evoke it first.

Overriding Constraints

The implemented version of ECG employs static constraint override using the keyword IGNORE. A constraint preceded by IGNORE is dropped. The analyzer assumes that a constraint marked by IGNORE was defined in a supertype, and then during the verification phase, that constraint gets removed from the constraint set for the subtype.

Naming Constructions, Schemas and Roles

The last five lines of figure 3.7 show how the names of identifiers (IDENT), slot chains (SLOTCHAIN), external types (EXTERNALTYPE), strings (STR), and probabilities (PROB) are defined.

- Identifiers are used to express the names of constructions, schemas, role and constituent names. An identifier has to start with a letter, but after that, it can have any number of letters, numbers, dashes, and underscores. Note that "." cannot be a part of an identifier because it would look like a slot chain.

- Slot chains (SLOTCHAIN) are repeated identifiers separated by periods. To disambiguate between an identifier and a slot chain, the SLOTCHAIN rule requires that there be at least one period in the slot chain. In the context free specification, there is an NonTerminal called "Var" that allows either an IDENT or a SLOTCHAIN.

- Types that belong to external type systems (EXTERNALTYPE) can be specified with the "@". The system makes few assumptions about the external type system's names, so they can be any combination of letters, numbers, periods, underscores and dashes without white space

- Strings (STR) are used to specify atomic, unstructured fillers of features. They are enclosed in double quotes and cannot be broken across lines or contain other double quotes (unless the " is preceded by a slash (e.g. \").

- Constituent probabilities (PROB) are expressed as any combination of digits preceded by a "." or "1.0" or "1".

3.3.1 Analyzer Grammar Constraints

The grammar writer must impose further structure on the grammar if he or she wants it to be compatible with the implemented analyzer. Specifically, the grammar writer must do the following:

- Lexical constructions are defined as concrete constructions without constituents and have a form pole with a feature orth. The orthographic form of the lexeme is assigned to the value of the orth feature using a ⟵ constraint in the form block.

- A special construction called Root must be defined. The analyzer knows that it has found a complete utterance when a completed instance of Root has been found, and no constructions can have Root as the type of one of their constituents. It is customary to define Root with a single constituent and allow that constituent to be filled by anything that counts as a reasonable utterance.

- If the grammar writer wants the analyzer to interact with context while using the grammar, the grammar writer must define an RD schema as defined in section 3.2.2. The information in the RD (the fillers of each of the roles) is used to find the referent corresponding to a reference.

- No role, constituent or evoked schema can be named "m" or "f".

3.3.2 Verifying the Grammar

A grammar that satisfies the context free specification is not necessarily a well-formed ECG grammar specification. There are many things that could still go wrong. For example, a type or a role might be undefined. This section describes the steps the pre-processor goes through to further verify the well-formedness of an ECG grammar.

For each construction and schema (referred to below as "type"), the pre-processor:

- Annotates each role, evoked item, and constraint with the construction or schema that originally defined it.

- In topological sort order starting from the root of the schema/construction hierarchy, inherited structure is accumulated and pushed along subcase links. During this process, name clashes are detected, and an error is signaled if the clashing names have incompatible types.[5] Name clashes can occur in two ways.

[5]Compatible here means that one of the types is a subtype of the other.

In one case, a type is a subcase of two parent types, both of which define the same role name. In the second case, a clash can occur between a parent type and a subtype if the subtype is redefining the type of a role. The system requires that the redefined type be a subtype of the original parent type.

- For each type, each of its constraints are checked to ensure that each slot chain role (with the exception of the final role in the slot chain) has a type and that the type contains the next role in the slot chain. More precisely, for the slot chain x.y.z, the code checks to make sure that the type associated with role x has a role y and the type associated with role y has a role z.

- Additionally, for co-indexation ($\longleftrightarrow$) constraints, the left hand slot chain and the right hand slot chain are checked to make sure that they have compatible pairwise (static) types.

Of course, the verification process can only check so much. Certain grammar errors can only be found during the analysis process. Debugging grammars during the analysis process is an area of future research.

3.4 Constructions as Feature Structures

Since ECG is a unification-based formalism, instances of constructions are represented as feature structures within the analyzer. For the most part, translating the role and equality constraints into feature structure notation is straightforward, with a few wrinkles to support the ignore operator and help reduce the size of the feature structures themselves.

Figure 3.8 shows the feature structure that corresponds to the MotionAlongAPath construction presented in figure 3.3 in the last chapter and duplicated here. The MotionAlongAPath construction in figure 3.8 describes VPs with a motion verb and a PP path specifier. Its corresponding feature structure captures all the information in the construction, but is intentionally underspecified with respect to the constituents. Underspecifying the structure of the constituents keeps the size of the parser states

somewhat smaller, allowing the system to store and copy less information. Additionally, well-motivated underspecification does not change the final structure of the semspec, and as will be shown below, it enables simple implementation of the ignore keyword.

The algorithm for generating a feature structure from a construction γ is as follows:

1. Add a root slot typed γ and filled with a feature structure. For the Motion-AlongAPath construction, this root slot is typed with the constructional type MotionAlongAPath.

2. Add an m role for the meaning pole, setting its type to the type of the γ's meaning pole. In our example, role m is typed as the schema type MotionAlongAPath.

3. If the constructional block is typed, add the constructional roles to the feature structure. In the example, we have added the role features which is defined in the constructional block type VerbFeatures.

4. Add typed roles for each of the constituents. In this case, feature v is typed to Verb and feature pp is typed to Spatial-PP.

5. Add typed roles for each of the evoked items in the meaning pole. In the example, feature ed is typed to schema EventDescriptor.

6. Look up the roles and evoked structures defined in meaning block's schema type. Add the roles underneath feature m. For example, the speed, heading, x-net, spg, protagonist, and mover roles defined in schema MotionAlongAPath are all added to the feature structure defined as the filler of m.

7. For each of the co-indexation constraints defined directly in the construction's meaning block and constructional block, the roles at the ends of the slot chains are co-indexed If a slot chain references a role that has not yet been added to the feature structure, then add the role and the appropriate type of that role. In our example, the self.m $\longleftrightarrow$ v.m
constraint is added in this manner, setting up the co-indexation represented as

construction MotionAlongAPath
subcase of VerbPlusArguments
constructional : VerbFeatures
constituents
v **:** Verb
pp **:** Spatial-PP
constraints
self.features ⟷ v.features
form
constraints
v.f before pp.f
meaning : MotionAlongAPath
evokes EventDescriptor **as** ed
constraints
self.m.mover ⟷ ed.profiledParticipant
self.m ⟷ v.m
self.m.spg ⟷ pp.m

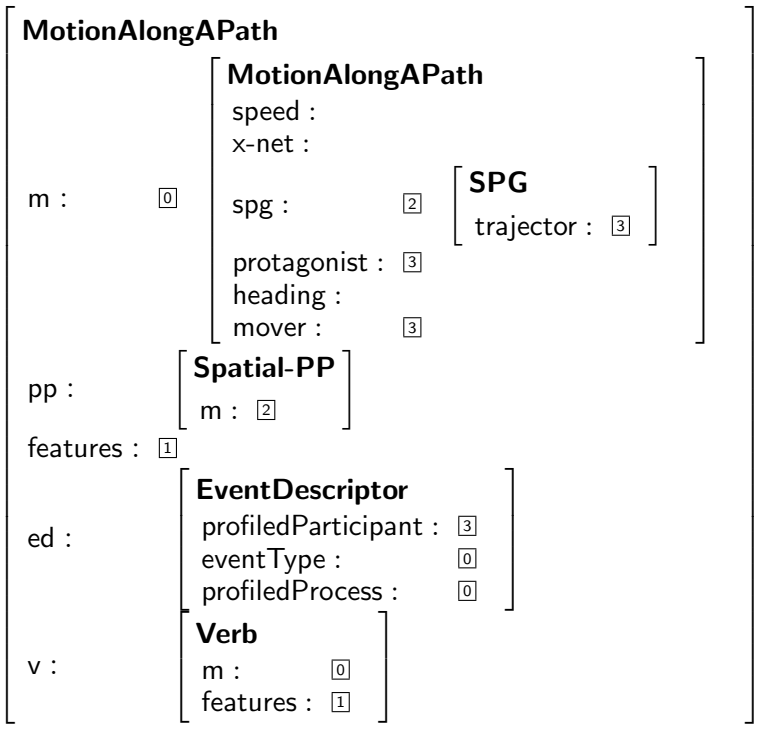

Figure 3.8: A MotionAlongAPath construction and its feature structure.

$\boxed{0}$. Incorporating self.m.mover $\longleftrightarrow$ ed.profiledParticipant
into the feature structure adds the profiledParticipant role to the EventDescriptor feature structure, and results in co-indexation $\boxed{3}$.

8. For each meaning constraint defined in the meaning block's schema type, co-index the roles at the ends of the slot chains (but precede each slot chain with m). If a slot chain references a role that has not yet been added to the feature structure, then add the role and the appropriate type of that role. In our example, the MotionAlongAPath's protagonist and mover roles are co-indexed in this manner sharing in co-indexation $\boxed{3}$.

3.4.1 Overriding in the Feature Structure

The ignore operator allows for a limited kind of constraint overriding. While constraint overriding can lead to nonmonotonic inference, the analyzer employs underspecification to keep from having to undo constraints during analysis.

Constraint overriding is implemented by just removing constraints marked with ignore from the construction's constraint list as well as any subtypes that would have inherited the constraint. Consider three types: D, E, and F. D defines constraint ρ, E overrides ρ with ignore, and F inherits the override, and so F does not have constraint ρ either.

Constraint overriding becomes a problem in inference when an inference engine makes certain assumptions about an instance i (e.g. birds can fly and can therefore travel quickly) only to later find out that the instance i's actual type is one that overrides some aspects of the supertype (e.g. a penguin).

One can cast this problem as type uncertainty, and the analyzer has type uncertainty with constructional constituents. A constructional constituent is defined as type t, but if actual filler type is t'. If constraint ρ holds on t, but not t', then the system cannot assume that ρ holds. Any assumption that the analyzer makes about t must be compatible with t'. But because the underspecifies constituent types, constraint ρ was never added in the first place. In short, underspecification in the feature structure is what makes this simple overriding scheme work.

Chapter 4

A Factored Model of ECG Analyses

This chapter describes the factored model used to model ECG analyses in detail. Because ECG is a construction grammar, the factors employed are syntactic and semantic. This chapter provides the motivation for using a factored model and shows how each factor is decomposed into the parameters used by the analyzer. The content of this chapter is conceptually dependent on chapter 2.

4.1 Motivation for a Factored Model

The goal of any parser is to find the highest scoring parse of a sentence. The constructional analyzer is no different in this regard. The analyzer searches for the analysis a that maximizes the conditional probability $\Pr(\, a \mid \mathit{sentence}, \mathit{grammar}, \mathit{context}\,)$ or the probability of an analysis given the sentence, the grammar and context.[1] Instead of working with the conditional distribution directly, I will instead define a generative model over analyses.[2], Using a generative model often makes it easier to

[1]The standard assumption is to treat the parse as independent of context, but I include it here for purposes which will soon become clear.

[2]Argmax of $\Pr(\, a \mid \mathit{sentence}, \mathit{grammar}, \mathit{context}\,)$ is the same as argmax of $\Pr(\, \mathit{sentence}, a \mid \mathit{grammar}, \mathit{context}\,)/\Pr(\, \mathit{sentence} \mid \mathit{grammar}, \mathit{context}\,)$ assuming that an analysis a generates the sentence with probability 1.

leverage the structure of the problem, which simplifies inference and parameter estimation.

While generative models are motivated by the structure of the problem and easy to understand as a result, building a generative model for an ECG analysis is not straightforward. ECG is a unification grammar, and as shown in chapter 2, ECG analyses are feature structures. Unfortunately, the limited work on defining a generative model over feature structures never properly dealt with the nonlocal dependencies inherent in feature co-indexation [4] [1].

The primary way researchers have addressed the problem of building a probabilistic model over feature structures is to go back to modeling

$$\Pr(\, a \mid sentence, grammar, context \,)$$

using discriminatively trained models. Log-linear models, for example, do not make independence assumptions about their features, and are applied by Johnson et al. [33] to unification-based grammars. Riezler [67] generalizes the work by Johnson et al. for German LFG. Further work by Kaplan et al. [66] shows that by applying log-linear models to the English LFG parser, one can achieve state of the art dependency extraction performance with the LFG parser.

The approach to probabilistic modeling of feature structures that I take in this dissertation is to decompose the feature structure into simpler probabilistic factors as is intuitively described in chapter 2. Given the lack of training data, using a factored model is a necessity, though this is not the only virtue of a factored model. Factored models are easier to understand and have fewer parameters than discriminatively trained models. Additionally, factored models are generally easier to perform incremental inference with because one can marginalize out as-of-yet unseen structure. This aspect is important since I am modeling incremental interpretation, and not using the model for re-ranking.

The best case for any factoring approach is when the factors require simple inference and parameter estimation, but still yield fast and accurate inference in the original domain when combined. Narayanan and Jurafsky use a factored model for their psycholinguistic results in which they model construction parsing as a mixture

of PCFG parsing, thematic role preferences, and n-gram models [57]. They assume independence between the factors, combining them with a noisy AND function. Klein and Manning use simple PCFG and lexical dependency factors for building a broad coverage parser of English [42].

To use a factored model for a problem, one decides which factors appropriately decompose the more complicated distribution and how those factors combine. Different methods for combining factors make different assumptions about the domain. For example, both Narayanan and Jurafsky and Klein and Manning assume independence between factors, though such an assumption is not strictly necessary. Haghighi et al. [27] show a method for combining factors when the factors are not independent.

4.2 Factoring an ECG Analysis

An ECG analysis consists of a set of interconnected construction instances τ and the semspec σ that those instances specify. Figure 4.1 (also shown in chapter 2) shows a DAG representation of an analysis of the sentence *he slid into the room*. Given the natural split of form and meaning information within a constructional analysis, factoring an analysis into a model of syntax and a model of semantics is an obvious step, and is broadly compatible with the work of Narayanan and Jurafsky [55] and Pado [60]. As you can see in figure 4.1, the syntactic and semantic factors are not independent because the choice of which filler is bound to which frame role depends crucially on the constructions employed in the analysis.

4.2.1 Defining the Factors

The probability of an analysis a given a grammar G and context Z is $\Pr(\, a \mid G, Z\,)$. As is shown in chapter 2, we can partition a with the variables τ representing the syntactic information in an analysis and σ representing the semantic information (or semspec) in the analysis.

$$\begin{aligned} \Pr(\, a \mid G, Z\,) &= \Pr(\, \sigma, \tau \mid G, Z\,) && (4.1) \\ &= \Pr(\, \sigma \mid \tau, G, Z\,) \Pr(\, \tau \mid G, Z\,) && (4.2) \end{aligned}$$

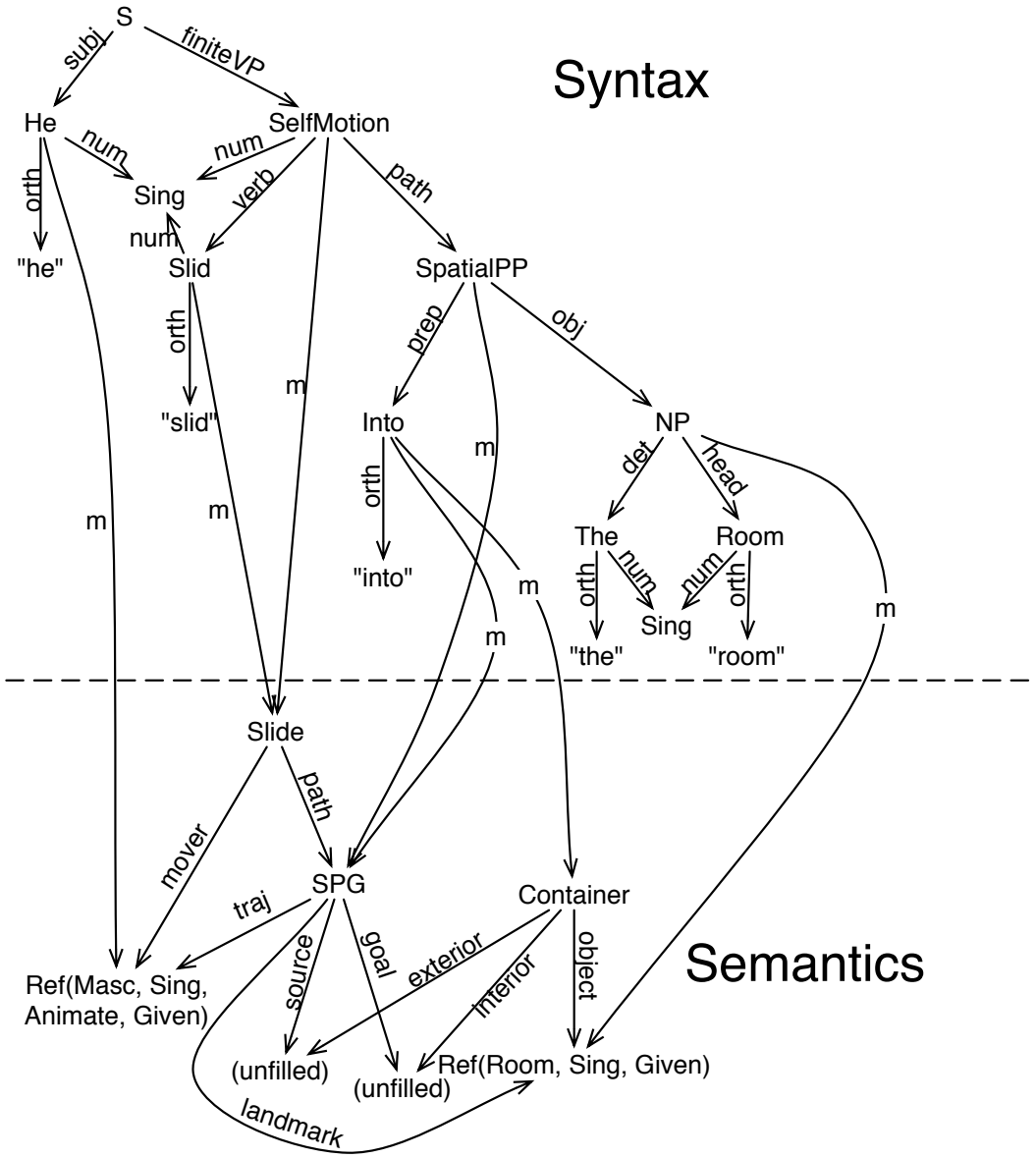

Figure 4.1: The DAG version of the feature structure shown in figure 2.1. Please see chapter 2 for a thorough discussion of this example.

$$= \Pr(\sigma \mid \tau, G, Z)\Pr(\tau \mid G) \quad (4.3)$$

Equation 4.1 defines the distribution over a in terms of the joint distribution over the semspec σ and the syntactic information τ. In both $\Pr(a \mid G, Z)$ and $\Pr(\sigma, \tau \mid G, Z)$, I assume that the words of the sentence are generated by a and τ, respectively. In equation 4.2, the chain rule is used to split up τ and σ. In equation 4.3, I assume that the probability of a τ does not depend on context. Obviously this independence assumption is problematic, but it is an assumption built into broad coverage parsers as well. Decomposing the probability of an analysis in this way shows how to factorize an analysis, and how those factors are combined.

4.3 Approximating $\Pr(\tau \mid G)$

I approximate the probability of factor $\Pr(\tau \mid G)$ with a generative context free model. Generative models of syntax calculate the probability of a tree t by breaking it down into the product of smaller syntactic choices conditioned on local information. In the simple PCFG model, for example, the probability of a tree is the product of the nonterminal expansion choices where each choice is only conditioned on the nonterminal being expanded [35]. However a generative model need not be so simple. The constructional factor employed in the analyzer is based on a model designed by Collins [11]. Collins' model is a generative model designed to parse into Penn Treebank [49] style trees.

In Collins' model, each syntactic category generates a subcategorization frames to determine its constituents. Syntactic subcategorization frames are related to constructions, and therefore his work is quite relevant. But instead of using subcategorization frames, the constructional analyzer uses the richer information provided by constructions. ECG Constructions specify:

- Which constituents are arguments and which are adjuncts
- Ordering constraints on constituents that admit partial orderings
- The likelihood of a constituent being omitted or expressed (locally or nonlocally)

- The allowable constructional filler types for each constituent

The parser's choice to use a particular construction in an analysis involves "sub"-choices to determine which of the construction's constituents are expressed and how they are expressed (locally or nonlocally). The ordering of the expressed constituents must be determined, and the actual filler of each constituent must be chosen as well. To make parameter estimation feasible, the likelihood of fitting a construction is broken down into each of these simple syntactic decisions conditioned only on local information.

4.3.1 Computing the Probability of a Construction

A construction in ECG specifies rich information about the construction, however there is still uncertainty in how the construction is realized. A constituent β can be expressed or omitted ($expressed_\beta$), local or nonlocal [3] ($local_\beta$), in a particular constituent ordering *ordering* and filled by a particular constructional filler $filler_\beta$. Thus the probability of the "RHS" of a construction α with $i+1$ constituents is:

$$\Pr(\, filler_{0\ldots i}, ordering_{0\ldots i}, local_{0\ldots i}, expressed_{0\ldots i} \mid \alpha\,) \tag{4.4}$$

Using the rules of conditional probability and a few reasonable independence assumptions, equation 4.4 can be rewritten as:

$$\begin{aligned} = \;& \Pr(\, filler_{0\ldots i} \mid expressed_{0\ldots i}, \alpha\,) \times \\ & \Pr(\, ordering_{0\ldots i} \mid local_{0\ldots i}, \alpha\,) \times \\ & \Pr(\, local_{0\ldots i} \mid expressed_{0\ldots i}, \alpha\,) \times \\ & \Pr(\, expressed_{0\ldots i} \mid \alpha\,) \end{aligned}$$

These independence assumptions state that the choice of constituent filler depends on whether the constituent is expressed and on the construction α. The choice of

[3] Collins employs a probabilistic model of gap threading in his generative model. More recent research by Levy [46] suggests that simpler methods of recovering gaps work better than Collins' generative model, and as a consequence, I am not encoding the full gap threading process into the model.

constituent ordering depends on which constituents are local and α. The choice of which constituents are local depends on which constituents are expressed and α, and that the choice of which constituents are expressed depends just on the construction α.

All of these independence assumptions are straightforward, however the term $\Pr(\, expressed_{0\ldots i} \mid \alpha\,)$ again calls into question the assumption made in equation 4.3 that the probability of a tree does not depend on context. With $\Pr(\, expressed_{0\ldots i} \mid \alpha\,)$, I assume that one can describe α's preference for expressing/omitting its constituents ignoring context.

Instead of making the decision to express/omit a constituent depend on context, I have encoded the the dependence in the semantic factor described below. Referents must be found for each of the omitted constituents, and constituents for which there are likely fillers given context pay a smaller penalty that those that are harder to resolve in context. Thus in the implemented system, whenever a constituent is omitted, it must be resolved from context for the construction to match.

For each of the terms except for *ordering*, I assume choices for each constituent are independent, and approximate them in the following way:

$$\Pr(\, expressed_{0\ldots i} \mid \alpha\,) \approx \prod_j \Pr(\, expressed_j \mid \alpha\,) \tag{4.5}$$

$$\Pr(\, local_{0\ldots i} \mid expressed_{0\ldots i}, \alpha\,) \approx \prod_j \Pr(\, local_j \mid expressed_j, \alpha\,) \tag{4.6}$$

$$\Pr(\, filler_{0\ldots i} \mid expressed_{0\ldots i}, \alpha\,) \approx \prod_j \Pr(\, filler_j \mid expressed_j, \alpha\,) \tag{4.7}$$

Equation 4.5 assumes that the choice of expressing or omitting a constituent does not depend on the choice of omitting/expressing other constituents. This assumption helps limit the number of parameters, but is problematic because it could assign non-zero probability to omitting all of the constituents. This caveat aside, Mok and Bryant [53] show that making this independence assumption leads to reasonable estimates for the omission patterns of the Chinese ditransitive construction. Additionally, the parsing strategy described in the next chapter will not use a construction without first matching a constituent.

Equation 4.6 states that the choices of whether the expressed constituents are

local do not depend on each other. As stated, this assumption is also problematic because it could assign a non-zero probability to all the constituents being nonlocal. However, the actual implementation conditions this choice on the existence of a fronted constituent, and a constituent can be nonlocal only if a fronted filler is available. This makes it impossible for more than one constituent to be nonlocal.[4] Additionally, should a construction choose for a constituent j to be nonlocal, the fronted constituent still acts as a filler for the $filler_j$ choice and thus also fills the appropriate semantic role.

Equation 4.7 assumes that the filler choice of a constituent only depends both on the construction and whether that constituent is expressed. Generative syntax models usually apply this assumption as it is consistent with context free assumptions. Conditioning on the construction (Klein and Manning [41] refer to this as conditioning on the parent) increases the accuracy of this estimate.

Various independence assumptions can simplify the computation and parameter estimation of the *ordering* variable. Collins' chooses the complete left and right orderings (subcat frames) in a single step using the Treebank data to infer the parameters. As is done by Klein and Manning [41], one could also treat constituent ordering as a Markov process. A first order process would condition the choice of which constituent would come in j^{th} position on the constituent in the $j-1^{th}$ position.

Regardless of which independence assumptions are employed, the parameters must be given or learned. Because the ordering parameters seem to be highly dependent on the construction, it is not obvious how to use existing training data like the Penn Treebank to estimate them. Also since the common case is a total ordering, I ignore this term in the implemented model[5]. However a future line a research could be inferring ordering parameters from linguistic resources.

[4]There are other kinds of nonlocal constituency besides fronting/topicalization/question nonlocality. To handle these cases, this variable must be split to distinguish between a constituent being expressed locally, expressed as a topicalized constituent, or expressed to the right of the current construction as a child of an ancestor in the tree. Also, the parser would have to allow nonlocal constituents to act as constituents of other constructions.

[5]Obviously one could use a uniform distribution over the remaining constituents, but that would unduly penalize constructions with many constituents.

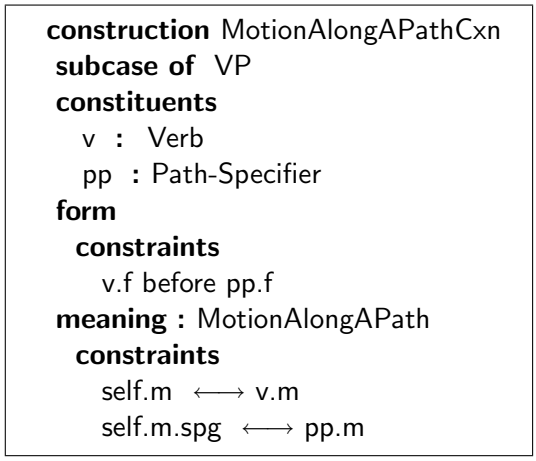

Figure 4.2: A simplified MotionAlongAPath construction. This construction matches the underlined portion of a sentence like *He slid into the room*.

4.3.2 Constructional Factor Example

Using the terms summarized in the previous section, we can compute the probability of the construction MotionAlongAPathCxn generating the past tense verb *slid* and destination phrase *into the room*:

$$\Pr(\, expressed_v \mid MotionAlongAPathCxn \,) \times$$
$$\Pr(\, local_v \mid MotionAlongAPathCxn, expressed_v \,) \times$$
$$\Pr(\, filler_v = SlidCXN \mid MotionAlongAPathCxn, expressed_v \,) \times$$
$$\Pr(\, expressed_{pp} \mid MotionAlongAPathCxn \,) \times$$
$$\Pr(\, local_{pp} \mid MotionAlongAPathCxn, expressed_{pp} \,) \times$$
$$\Pr(\, filler_{pp} = DestPhrCxn \mid MotionAlongAPathCxn, expressed_{pp} \,)$$

In this case, the probability that the verb v is expressed and the probability that the verb is local are multiplied by the probability that the constituent filler of v is the SlidCxn. Similarly, the expressed, locality and filler probabilities are multiplied in for the *pp* constituent. If the *pp* constituent had been omitted, $\Pr(\, omitted_{pp} \mid$

$MotionAlongAPathCxn$) would have been used instead, and

$$\Pr(\, local_{pp} \mid MotionAlongAPathCxn, expressed_{pp}\,) \times$$
$$\Pr(\, filler_{pp} = DestPhrCxn \mid MotionAlongAPathCxn, expressed_{pp}\,)$$

would not have been included.

If the *pp* constituent had instead been been fronted, as in the sentence, *Into the room he slid*, $\Pr(\, nonlocal_{pp} \mid MotionAlongAPathCxn, expressed_{pp}\,)$ would have instead been used, but the same constituent filler probability would have been included because the MotionAlongAPathCxn and the fronting construction would both be responsible for the destination phrase (although probabilistically, they are treated by the analyzer as unrelated clauses of the constituent).

4.3.3 Constructional Factor Summary

Assuming construction α, constituent β and filler type θ, the probabilistic syntactic parameters employed in the model are:

- $\Pr(\, expressed_\beta \; vs. \; omitted_\beta \mid \alpha\,)$: This is the probability that constituent β is expressed or not expressed (omitted) given the construction. Note that this term does not depend on the context, as it models a construction's preference for omitting an argument ignoring context, letting reference resolution incorporate context.

- $\Pr(\, local_\beta \; vs. \; nonlocal_\beta \mid \alpha, expressed_\beta\,)$: The probability of construction α's constituent β being expressed locally (as a normal constituent) or expressed nonlocally as a fronted constituent

- $\Pr(\, filler = \theta \mid \alpha, expressed_\beta\,)$: Assuming β is expressed, this term is the likelihood that a particular constructional type θ fills constituent β.

4.4 Computing $\Pr(\sigma \mid \tau, G, Z)$

With a method for estimating $\Pr(\tau \mid G)$ in hand, I will now show how I approximate the probability of the semspec σ given the syntactic information in τ, the grammar G and the context Z. As I argue in chapter 2, the information in a complete τ deterministically specifies the structure of σ. However the likelihood of the nonlocal dependencies in σ have not been taken into account in $\Pr(\tau \mid G)$, so they must be scored by the semantic factor.

Figure 4.3 shows the semspec associated with *he slid into the room* represented as a DAG. Three different kinds of information are stored in the semspec. The first is the network of arcs representing the set of role sets in the DAG that are co-indexed. For example, the mover and traj roles are one such co-indexed set, the object and landmark roles are another, and the path role is another.[6] The second kind of information is hidden in the Ref(...) function. The Ref function finds a referent for a given set of features such as masculine, singular animate, and given. The final kind of information in the semspec are the role=filler pairings. For example, the fact that the referent of Ref(Masc, Sing, Animate, Given) is what is filling the mover role and the trajector role.

In the following equations, I use a different variable for each kind of information. I call the set of role sets that are co-indexed together *csets*. I call each referent returned by the Ref function a *filler* and the set of fillers across multiple Ref calls *fillers*. And the assignments of fillers to roles I call *assignments*. Using these new variables, we can decompose the likelihood of a semspec σ:

$$\begin{aligned}
\Pr(\sigma \mid \tau, G) &= \Pr(assignments, fillers, csets \mid \tau, G, Z) && (4.8)\\
&= \Pr(assignments \mid fillers, csets, \tau, G, Z) \times \\
&\quad \Pr(fillers \mid csets, G, Z) \Pr(csets \mid \tau, G, Z) && (4.9)\\
&\approx \Pr(assignments \mid fillers, csets) \Pr(fillers \mid csets, \tau, G, Z) \times \\
&\quad \Pr(csets \mid \tau) && (4.10)
\end{aligned}$$

Equation 4.8 substitutes the appropriate variables for σ. Equation 4.9 uses the chain

[6] The sets can be of size one.

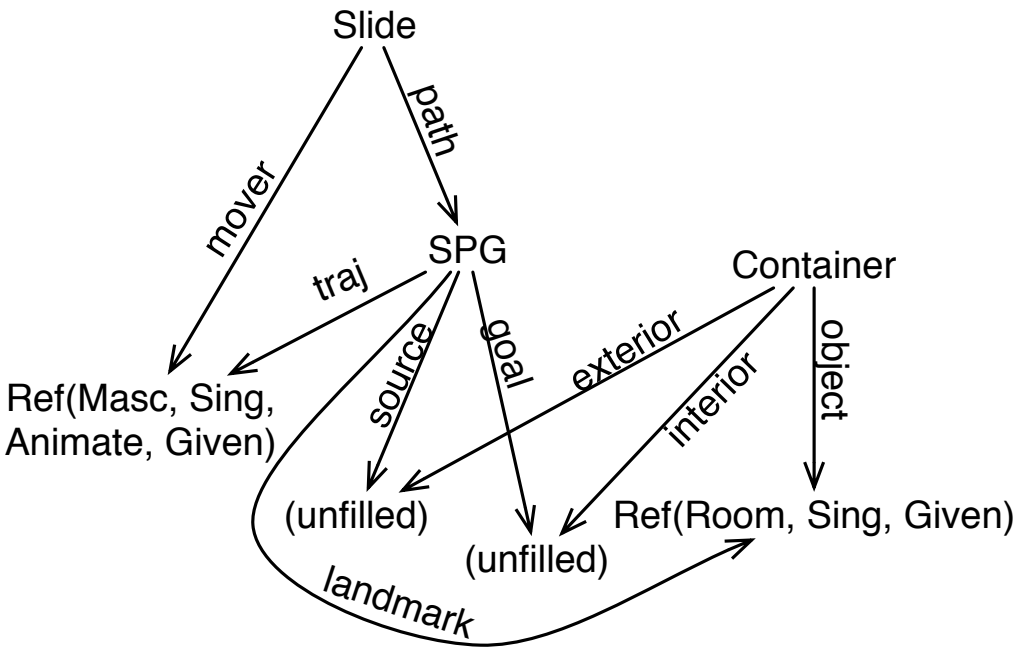

Figure 4.3: The semantic partition of the feature structure for *he slid into the room.*

rule to split up the variables. Equation 4.10 makes an independence assumption that *assignments* only depend on the *roles* and the $fillers$ and do not depend on τ, the grammar G or the context Z. In essence, this independence assumption states that given the *csets* and the $fillers$, the semantic factor has enough information to evaluate the quality of the semspec in terms of consistency between the roles and their fillers.

To summarize, the terms of equation 4.10 represent:

- $\Pr(\, csets \mid \tau, G\,)$ represents the likelihood of a set of co-indexation sets given the constructions used in an analysis. Or in terms of the DAG formulation, the choice about which arcs point to the same thing. For the example semspec in figure 4.3, this term computes

$$\Pr(\, \mathsf{mover} = \mathsf{trajector}, \mathsf{object} = \mathsf{landmark}, \mathsf{path} \mid \tau\,)$$

- $\Pr(\, fillers \mid csets, \tau, G, Z\,)$ represents the likelihood of a particular set of referent assignments over all the Ref calls in the semspec. For the example semspec, this term computes

$$\Pr(\, Ref(Masc...) = Joe, Ref(Room...) = Room42 \mid \tau, csets, G, Z\,)$$

- $\Pr(\, assignments \mid fillers, csets\,)$ computes the likelihood of each filler being paired up with its cset. For the example semspec using the example referents "J" for "Joe" and "R" for "Room42", it computes:

$$\Pr(\, mover = J, traj = J, lm = R, obj = R, path = SPG \mid J, R, csets\,)$$

4.4.1 More on $\Pr(\, csets \mid \tau, G\,)$

Given the complete constructional information in τ, there is no uncertainty about the set of co-indexed roles associated with τ because the constructions employed in τ are explicit about which roles are unified. Thus $\Pr(\, csets \mid \tau, G\,) = 1$ for a complete τ. But if τ were only partially specified (as is the case during parsing), there would be uncertainty about the structure of the semspec, and $\Pr(\, csets \mid \tau, G\,)$ takes on

an important role. As is shown in chapter 2, a partial tree τ could yield a set of semspecs, and thus $\Pr(\, csets \mid \tau, G\,)$ trades off the likelihood for each viable semspec structure given τ.

One can either treat $\Pr(\, csets \mid \tau, G\,)$ as:

- Option 1: a distribution over semspec structures (i.e. different sets of csets)
- Option 2: The combination of many smaller distributions determining whether $role_i$ and $role_j$ are co-indexed.

For option 1 the number of semspec structures in the worst case is equal to the number of different role subsets (i.e. intractable). For option 2, the data structures and inference must be sophisticated since it boils down to a problem of identity uncertainty over roles. Because option 1 is easier to explain, understand, and do inference on, the implemented system uses uncertainty over complete semspec structures as a model of $\Pr(\, csets \mid \tau, G\,)$. As shown in the next chapter, computing $\Pr(\, csets \mid \tau, G\,)$ in this way turns out to be easy because a partial τ does not yield many semspecs in practice.

4.4.2 More on $\Pr(\, fillers \mid csets, t, G\,)$

In an ideal case, a semantically rich reference resolution model would be used to model $\Pr(\, fillers \mid csets, t, G\,)$. This factor would take facts about a partial analysis, employ a model of discourse and situational context, and return a ranked set of referents for each utterance. Building this ideal reference resolution factor was an early goal of this research.

Unfortunately, current theoretical models don't support incrementality, are not psychologically plausible, and are not structured in a cognitively motivated way. Given the lack of theoretically clean accounts of reference resolution that satisfy this dissertation's desiderata, I implemented a heuristic model of situational and discourse context, which could one day be the basis for an ideal reference resolution factor. The situational and discourse context model takes facts about a reference, such as gender, number, givenness and semantic type and returns a ranked set of candidate referents.

The analyzer uses a heuristic to return a list of candidates for each reference along with the constructional tree and the semspec. For the semantic factor, instead of using the type of the referent to compute the semantic cost, I used the head noun of an NP as a proxy for the referent type.

The related work on discourse and situational context modeling broadly falls into two categories: broad coverage computational work and linguistic models based on logical inference. Successful approaches to coreference resolution in the natural language processing literature do not use structured models of discourse and situational context, but instead use pairwise matching functions [77], [58]. It has been shown that pairwise matches can be further refined with clustering methods on top of the pairwise methods [50]. While a pairwise matching solution is compatible with incremental processing, it is a a mismatch to a structured model of discourse and situational context, and therefore hard to apply in the context of this dissertation.

Recent NLP research in unsupervised coreference resolution by Haghighi and Klein [28] uses a generative model of document discourse. In their model, the entities in the document are generated by an infinite mixture model, and then each entity generates the mention found in the document. While their intriguing work might form the basis of a theoretically sound version of a reference resolution factor, turning their work into a cognitive model is beyond the scope of this dissertation.

On the other end of the spectrum are structured linguistic models such as Discourse Representation Theory DRT [36]. While compatible with the idea of a structured model of context, DRT does not provide a well-defined notion of referent ranking. As a consequence, such linguistically insightful theories are not easily usable to define a reference resolution factor. Additionally, recent work in applying DRT to Chinese ellipsis [73] seems no more theoretically sound than my ad hoc implementation.

4.4.3 $\Pr(\, assignments \mid fillers, csets \,)$

The goal of $\Pr(\, assignments \mid fillers, csets \,)$ is to determine the likelihood of role=filler bindings in a semspec, assuming a fixed structure (i.e. a complete set of

co-indexations) and a given set of fillers. Thus this factor can be thought of as a probabilistic implementation of traditional selectional restrictions [35].

Gildea's work on automatic frame role labeling [24] ignited a flurry of research in probabilistic models of frame role fillers. Some of the best performing systems have been feature-based SVM systems [64] that use semantic features and syntactic features that map the parse tree into a role/filler set. Clustering over framenet frames, Pado has built an interesting joint (generative) model over fillers, roles and frames that she has used to mimic human semantic judgments [59].

4.4.4 Computing $\Pr(\, assignments \mid fillers, csets \,)$

Given the fillers and the coindexations between roles (i.e. the structure) of a semspec, $\Pr(\, assignments \mid fillers, csets \,)$ can be computed in the following manner:

$$\Pr(\, assignments \mid fillers, csets \,)$$

$$\approx \prod_{filler} \Pr(\, assignments_{filler} \mid filler, cset_{filler} \,) \tag{4.11}$$

$$\approx \prod_{filler} \Pr(\, f_0.role = filler, \ldots, f_k.role = filler \mid filler, cset_{filler} \,) \tag{4.12}$$

$$\approx \prod_{filler} F(\Pr(\, role_{f_0} \mid filler, f_0 \,), \ldots \Pr(\, role_{f_k} \mid filler, f_k \,) \tag{4.13}$$

In equation 4.11, each of the fillers gets treated as independent. Obviously this assumption is problematic given that knowing one participant in a frame is informative about the other participants in the frame. For example, knowing the frame is Eat and the agent is a Person leads to different guesses about the Eaten-Item than if the agent were of type Horse. However this assumption is fairly standard [59], [57]. Equation 4.12 decomposes a filler's assignments into a joint distribution over the role=filler pairs with the same filler. The joint effect of these pairs is approximated by equation 4.13 in which the fit of a filler to its assigned roles is decomposed in terms of a potential function F. Function F evaluates the combined effect of the filler being bound to that particular set of roles. Figure 4.4 illustrates the effect of these independence assumptions on th example semspec for *he slid into the room*

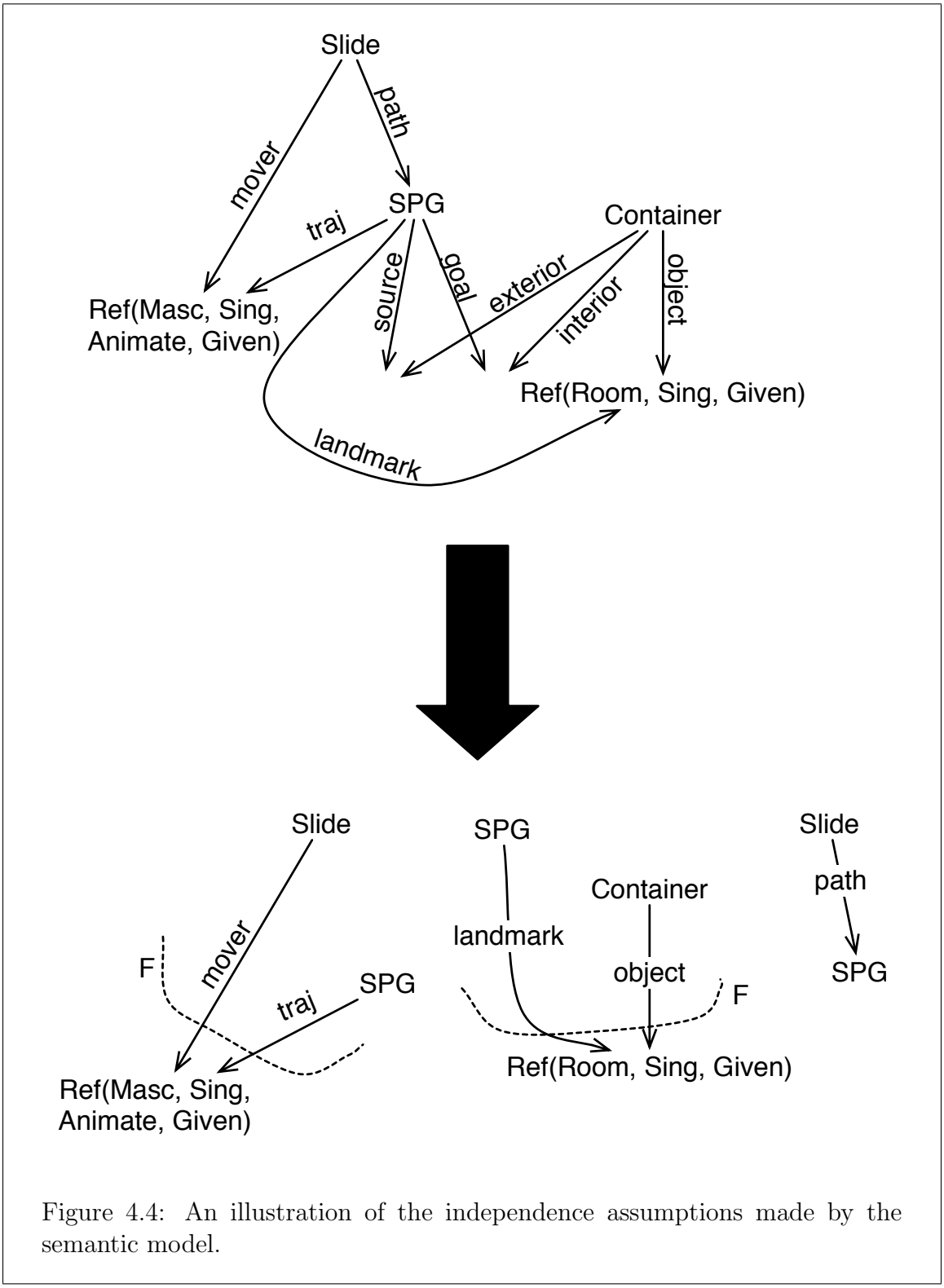

Figure 4.4: An illustration of the independence assumptions made by the semantic model.

As explained in chapter 2, I use function F because I do not want to treat each role as independent. Treating each role as independent would punish grammar rules with rich semantic information, and thus is contrary to the goals of this work. In an ideal situation, function F would be a learned as a log-likelihood function and thus maintain proper probabilistic semantics. However, since I do not have training data, I implement F as the harmonic mean, averaging the effects of each of the role=filler pairings.

4.5 Parameter Estimation Issues

Without training data, estimating the necessary parameters for the syntactic and semantic factors is problematic. In fact, with no training data and little testing data, a more fundamental question is how one can decide whether the model has too many or too few parameters, and whether the application of independence assumptions was too liberal.

Without training data, there are no empirical answers to these questions. The best that can be done is to apply the system to many problems to see which parameters matter. While parameter estimation issues and choices will be covered thoroughly in the case studies, this section provides a brief summary of how the parameters were estimated.

For the case study in chapter 6 that describes how the analyzer is used to model linguistic phenomena, the qualitative performance of the system is more important than any quantitative metric. Thus uniform distributions are used for the constructional filler preferences, constituents are expressed/local with probability one, and no semantic parameters were employed.

For the reading time work in chapter 7, constituents were expressed/local with probability one, but constructional filler probabilities were treated as context free[7] and inferred from Treebank rules. The semantic role=filler parameters are taken from McRae et al. [51], and are crucial for modeling the data.

[7]In this case, a constituent's filler does not depend on the construction, only on its type.

For the parsing of child-parent interactions in Mandarin described in chapter 8, the full set of parameters is employed. This case study also has a small amount of training and testing data, and thus a machine learning method in which all the parameters were learned from the data is employed.

Chapter 5

Left Corner Construction Parsing with a Factored Syntax-Semantics Model

This chapter describes a probabilistic left corner parsing algorithm applied to parsing with constructions defined in the Embodied Construction Grammar formalism [3]. The probabilistic model guiding the search is a factored model over syntax and semantics. The left corner parsing algorithm [70] [15] is chosen both because it has been shown to be efficient for parsing unification grammars [86] and because variations on left corner parsing have been shown to be psychologically plausible [78].

This chapter first provides a bit of background information on parsing strategies, search heuristics, parsing unification grammar and psychologically plausible parsing. In section 5.2, the probabilistic version of the left corner parsing algorithm for probabilistic context free grammars is defined. Then section 5.4 shows how the algorithm is extended for parsing using ECG as the grammar formalism.

While reading this chapter, keep the goals of the dissertation in mind. The constructional analyzer is both a natural language processing system as well as a cognitive model of language analysis. Therefore it must provide rich linguistic analyses using constructions, act as a semantic analyzer for small-to-medium-sized applications, and analyze sentences in a way that consistent with experimental data. Many systems

can do two out of the three tasks, but none that I know of can effectively do all three.

5.1 Parsing Overview

A parsing algorithm is a search process that takes a linguistic form (sentence) as its input and returns a set of hidden structures associated with that input. In syntactic parsing, this hidden structure is called a parse tree, but in joint syntax and semantic parsing, the hidden structure is a (partial) interpretation of the input form. This section provides a *brief* introduction to parsing strategies, search heuristics, parsing unification grammar and psychologically plausible parsing. With this background in place, one can explain why the left corner parsing strategy is an appropriate choice for a cognitive model of sentence processing.

5.1.1 Parsing Strategies

One way to distinguish parsing strategies is the order in which they attempt to build phrases. Bottom-up strategies start with the words and build up the smaller phrases before trying to build up larger ones. The primary concern with bottom-up strategies is that they are not goal directed. Words and phrases are grouped without taking into account the fact that the parser is trying to build a single consistent grouping for the whole sentence. Most high-performance broad coverage syntactic parsers employ a bottom up strategy [11] [10] [62].

On the other end of the spectrum, top-down strategies start with the assumption that a parser is looking for a consistent grouping of the whole sentence. Instead of grouping words into phrases and phrases into larger phrases, a top down parser starts with a proposed global grouping, and breaks it apart into its constituent phrases. Those phrases are decomposed into smaller phrases, and then hopefully the smaller phrases are re-written as the words of the sentence. The primary concern with top down parsers is that they are not data directed. Pure top down parsers do not condition their choices on the words of the sentence, and generally they are not used

for natural language parsing.[1]

Mixed strategies combine top-down expectations with bottom up filtering, or (viewed in the other direction) build only the bottom up phrases that are consistent with a top down expectation (or goal). The Earley parsing algorithm is a top down algorithm with bottom up filtering [17], while left corner parsing is a bottom up algorithm that only builds phrases if they are consistent with a top-down goal [15]. Recently, mixed strategies have received attention in the broad coverage syntactic parsing community [13] [48] [31], often with the goal of applying the mixed strategies to language modeling [68] [88]. The psycholinguistics community has also embraced mixed strategies because they are considered to be more psychologically plausible than pure top-down or bottom-up strategies [68] [60] [29] [46].

One other degree of freedom for parsing strategies is the order in which they process the words of the sentence. Incremental parsing strategies start by integrating the first word of the sentence, and then the next word and so on. Mixed parsing strategies are generally incremental, but the most commonly used parsing systems do not process the input incrementally.

5.1.2 Heuristic Parsers

Natural language is ambiguous, and depending on your assumptions about grammar, natural language can be extremely ambiguous[2]. With simple context free assumptions, the number of parses grows exponentially with the length of the sentence. That is too many parses, and so various heuristics have been developed to improve the speed and accuracy of syntactic parsers.

The most common heuristic used by broad coverage parsers is to treat the grammar as a probabilistic (or weighted) grammar. Probabilistic grammars provide a ranking

[1] According to the classification I am using, Roark [68] is a mixed strategy since the choice of action for his top-down parser is conditioned on the words.

[2] A richer grammar formalism like HPSG admits (on average) a much smaller set of parses per utterance [87] (on the order of 10), however, the grammar does not cover as much as a treebank-based grammar, so it is hard to compare these numbers. Additionally, parsers for richer grammar formalisms must do more work per analysis than a treebank parser, so a heuristic ranking function is just as important for a richer grammar.

function over parse trees, and the parser can focus on building promising candidates. The previous chapter describes various approaches to building a (probabilistic) ranking function over parse trees and interpretations. Importantly, the choice of ranking function depends not only on mathematical issues such as parameter estimation and ease of computation, but also on the choice of parsing algorithm.

For a ranking function to be effective, the conditioning information used to rank parses must be available in the parser. For example, bottom up and top down parsers share locality assumptions with PCFGs,[3] and thus a PCFG estimate is a natural ranking function for these parsers. Broad coverage systems that condition on nonlocal information must rely on additional search heuristics such as figures of merit [6] [13], A^* heuristics [40] [27] or post-parsing re-ranking functions such as [12], but these approaches are external to the probabilistic grammar.

For parsers that use mixed strategies, such as the Earley parsing algorithm or the left corner parsing algorithm, the locality assumptions made by a PCFG are different than the notion of locality used by the parsing algorithm. Mixed parsing strategies alternate bottom-up and top-down operations, so these strategies have access to top-down conditioning information as well as bottom-up conditioning information. Thus the probability of a parser state in a mixed strategy does not necessarily equal the probability of any particular (connected) syntactic structure. Care must be taken when relating probabilistic grammars to mixed parsing strategies because of this mismatch.

Stolcke [80] shows how to define the probability of each parser state in an Earley parser in terms of a PCFG grammar, while Manning and Carpenter [48] (and later Van Uytsel [88]) define a completely different probability model for left corner parsing that does not rely on PCFGs. Henderson uses a recurrent neural network approach for estimating the probability of a left corner parser state [31] [45], while Titov and Henderson use incremental sigmoid belief networks to estimate the probability of a left corner parser state [84]. Stolcke's stochastic Earley parser and Manning and Carpenter's probabilistic left corner parsing model (described in more detail in section

[3]Presumably this is because most parsing algorithms are designed to parse CFGs.

5.3.1) inform the approach taken here.

5.1.3 Parsing Unification Grammar

Unification grammar is more expressive than context free grammar, and in the worst case, recognition with unification grammar is intractable [86]. Tomuro, in her dissertation on left corner unification parsing [86], explains that most unification-based systems don't worry about this theoretical bound, and just treat unification grammar parsing as an extension to context-free grammar parsing. For example, Shieber's original unification parsing algorithm is just an extension of the Earley parsing algorithm [86].

Tomuro defines a left corner parsing algorithm for unification grammar that is sound and complete [86]. The technique I define in section 5.5.1 to build up a coherent semantic interpretation given a partial constructional tree is quite similar to the technique she uses to build coherent feature structures for a partial unification grammar analysis[4]. Tomuro tests her highly optimized left corner parsing algorithm on 125 sentences from a limited domain and observes that her algorithm produces 30% fewer parser states than Shieber's algorithm. This suggests that left corner parsing not only is a sound and correct method for parsing with unification grammar, but also an efficient algorithm for parsing unification grammar.

5.1.4 Cognitive and Psychological Plausibility

A cognitively motivated model of parsing and interpretation imposes additional constraints on the parsing task. For example, the traditional pipeline architecture where the syntax of an utterance is processed prior to the utterance's semantics is not cognitively viable because thematic-fit constraints have been shown by psycholinguists to affect incremental processing of a sentence [51]. Indeed, psycholinguistics has a lot to say about human processing, and any cognitive model of processing must also be plausible by psycholinguistic standards.

[4]Although if I understand her algorithm, my technique is much more efficient in terms of space and processing time.

But what does "psychologically plausible" mean? The most basic notion is that a psychologically plausible model generates predictions consistent with experimental evidence. In addition to this performance metric, the psycholinguistics community has proposed processing desiderata for models that claim to be psychologically plausible. Pado [59] suggests that psychologically plausible parsers must:

- Process the input incrementally
- Incorporate statistical information about the linguistic domain. Probabilistic models, for example, can mimic some of the experiential basis of language processing.
- Utilize limited parallelism in terms of the number of parses that can be simultaneously maintained.
- Condition parser choices on semantics. Psycholinguistic studies have shown that semantics affects incremental reading time [51].

Roark's [68] parser is an example of a parser that satisfies some of these desiderata. He built a broad-coverage, incremental, mixed-strategy syntactic parser that links each incoming word to a fully-connected in-progress parse tree. He uses a grammar inferred from the Penn Treebank that has undergone a transformation to remove left recursion. While Roark never used his parser to generate reading time predictions, Pado [60] uses it in her psycholinguistic modeling work. More of Pado's work will be covered in chapter 7.

5.1.5 Probabilistic Left Corner Parsing Using a Factored Syntactic and Semantic Model

The left corner parsing algorithm has been shown to be an efficient algorithm for parsing unification grammar [86]. It is an incremental algorithm that can take advantage of statistical information [48] [88] [31], employ limited parallelism, and, as will be shown, the parser actions can be conditioned on semantic information. Thus left corner parsing can be used to model language interpretation in an efficient and

psychologically plausible way using a rich, unification-based language formalism. In short, it is well-suited to act as a cognitive model.

In this chapter, I will first define the left corner parsing algorithm for CFGs, and then derive a probabilistic left corner parsing algorithm for PCFGs. This derivation provides insight on extending the algorithm to ECG. Then the probabilistic left corner parsing algorithm is redefined in terms of the factored syntactic and semantic model from the previous chapter, first for the syntactic factor, and then for the semantic factor.

The previous chapter ignored parsing issues and defined the probability of an analysis a as the product of the following terms:
$\Pr(\, a \mid G\,) \approx$

$$\Pr(\, assignments \mid fillers, csets\,) \Pr(\, fillers \mid csets, \tau, G, Z\,) \Pr(\, csets \mid \tau\,) \Pr(\, \tau \mid G\,) \tag{5.1}$$

Where $\Pr(\, \tau \mid G\,)$ is the probability of the syntactic information in the feature structure given a grammar G, $\Pr(\, csets \mid \tau, G\,)$ is the probability of set of co-indexed role sets $csets$ (semspec structures) given τ, $\Pr(\, fillers \mid csets, \tau, G, Z\,)$ is the probability of a set of referent fillers for each of the referring expressions given the τ and $csets$, and $\Pr(\, assignments \mid fillers, csets\,)$ is the probability of the role-filler pairings in the semspec.

Left corner parsing will provide the framework for conditioning the syntactic factor on the words. Instead of calculating the joint probability of $\Pr(\, \tau, words \mid G\,)$, the algorithm calculates $\Pr(\, t \mid G, words\,)$. Crucially, the construction-conditioned terms from the generative decomposition of $\Pr(\, \tau \mid G\,)$ are still used to compute $\Pr(\, \tau \mid G, words\,)$. Thus the model is still generative, but the parser uses Bayes rule to condition the inference on the words.

5.2 The Left Corner Parsing Algorithm

The left corner parsing algorithm [70] [15] conditions parser operations both on top down information (expected or "goal" categories) and bottom up information (words).

A left corner parser starts with a top down expectation of finding a complete sentence that begins with the first word of the input utterance. It is an incremental algorithm, so it only tries to incorporate word w_i if word w_{i-1} has already been derived. A new grammar rule r is used in the derivation only if it is consistent with the top down expectation and r's left-most constituent has already been derived from the input. Then r's next unmatched constituent becomes the top-down expectation for the next unincorporated word. Complete, derived subtrees can also be directly attached to the unmatched constituent (the top down expectation), if they are compatible.

Left corner parsers generally use a stack to represent each in-progress parse. The stack is made up of an ordered set of partially matched grammatical rules which I will call items. The top of the stack I_{top} is the most recently added item (partially matched grammatical rule). The ordering on the stack is indicative of syntactic dominance. The bottom of the stack is the root item of both the stack and of the parse tree. The item I_{top-1} immediately below the item at the top of the stack must syntactically dominate the top of the stack meaning that I_{top} must either be a child of I_{top-1} or a child of a descendant of I_{top-1}.

In this chapter, items on the stack will be represented as a grammar symbol along with its leftover (unmatched) constituent expectations (goals) in brackets[5]. Assuming that α, β, γ are variables standing for symbols of the grammar, then $\alpha[\beta\gamma]$ is a stack state originally pushed onto the stack using a rule like $\alpha \rightarrow \ldots \beta\gamma$, and a stack with j states would look like $\alpha_0[\beta_0\gamma_0], \alpha_1[\beta_1\gamma_1], \ldots, \alpha_j[\beta_j\gamma_j]$ where α_0 is the root of the stack and α_j is the top of the stack.

Left corner parsers build up the parse tree using three stack operations:

- **Push:** Pushes the current word θ onto the stack.

$$\alpha_0[\beta_0\gamma_0], \ldots, \alpha_j[\beta_j\gamma_j] \textbf{ becomes } \alpha_0[\beta_0\gamma_0], \ldots, \alpha_j[\beta_j\gamma_j], \theta[]$$

  The top of the stack must be incomplete, and there must exist[6] a path in the grammar between β_j and θ.

[5] An empty pair of brackets indicates a complete state.

[6] Unless the grammar allows for productive omission, which is also handled by the implemented model, but is covered in a later section.

- **Attach:** Remove the complete top of the stack, assigning it to be the left-most unmatched constituent of its stack ancestor. For example, assume that $\beta_{j-1} = \alpha_j$ then:

$$\alpha_0[\beta_0\gamma_0], \ldots \alpha_{j-1}[\beta_{j-1}\gamma_{j-1}], \alpha_j[] \textbf{ becomes } \alpha_0[\beta_0\gamma_0] \ldots \alpha_{j-1}[\gamma_{j-1}]]$$

- **Propose new item α' at j:** Given a rule $\alpha' \rightarrow \lambda'\beta'\gamma'$, assume that the complete α_j on the top of the stack matches λ'. Pop $\alpha_j[]$ and push a new state:

$$\alpha_0[\beta_0\gamma_0], \ldots \alpha_{j-1}[\beta_{j-1}\gamma_{j-1}], \alpha_j[] \textbf{ becomes } \alpha_0[\beta_0\gamma_0], \ldots \alpha_{j-1}[\beta_{j-1}\gamma_{j-1}], \alpha'_j[\beta'_j\gamma'_j]$$

Figure 5.1 shows an example application of the parser operations on the sentence *he slid the box.*

5.3 Probabilistic Left Corner Parsing

A probabilistic left corner parsing algorithm assigns a probability to each (partial) parse τ_i that is consistent with the input through word i. The probabilistic definition provided in this section combines ideas from Stolcke's probabilistic Earley parser [80] and Manning and Carpenter's probabilistic left corner parser [48].

5.3.1 Related Probabilistic Algorithms

Both probabilistic bottom-up and top-down parsers share locality assumptions with PCFGs. Each generated subtree multiplies the probability of a rule by the probability of each generated subtree, conditioned on information local to the phrase type.[7] Thus the (inner) probability of a subtree is easy to calculate because the parser's actions equate to the subtree generation decisions in the PCFG.

During the derivation of a parse tree, the left corner parsing algorithm maintains multiple incomplete subtrees (in a dominance relation) on a stack. To propose rules that are consistent with the stack, the left corner parser actions must be conditioned

[7]Technically, a bottom-up parser builds the subtrees first, but the same generative story holds.

- $Root[S]$
  - Push $HePronoun \rightarrow he$
- $Root[S], HePronoun[]$
  - Action: Propose $Decl \rightarrow Subj\ FiniteVP$
- $Root[S], Decl[FiniteVP]$
  - Action: Push $SlidePastTense \rightarrow slid$
- $Root[S], Decl[FiniteVP], SlidePastTense[]$
  - Action: Propose $Trans \rightarrow Verb\ NP$
- $Root[S], Decl[FiniteVP], Trans[NP]$
  - Action: Push $TheDet \rightarrow the$
- $Root[S], Decl[FiniteVP], Trans[NP], TheDet[]$
  - Action: Propose $DetNoun \rightarrow Det\ N$
- $Root[S], Decl[FiniteVP], Trans[NP], DetNoun[N]$
  - Action: Push $BoxN \rightarrow box$
- $Root[S], Decl[FiniteVP], Trans[NP], DetNoun[N], BoxN[]$
  - Action: Attach $BoxN$ to $DetNoun.N$
- $Root[S], Decl[FiniteVP], Trans[NP], DetNoun[]$
  - Action: Attach $DetNoun$ to $Trans.NP$
- $Root[S], Decl[FiniteVP], Trans[]$
  - Action: Attach $Trans$ to $Decl.FiniteVP$
- $Root[S], Decl[]$
  - Action: Attach $Decl$ to $Root.S$ and it's complete

Figure 5.1: The steps a left corner parser might go through for *he slid the box*

on information that is not local to any particular phrase type, but rather depends on features of the stack itself. Thus relating a probabilistic grammar with certain locality assumptions with a parsing algorithm with different locality assumptions requires some subtlety.

For example, instead of computing the probability of a partial tree directly, Stolcke's probabilistic Earley parsing algorithm [80] and Manning and Carpenter's probabilistic formulation [48] of left corner parsing both calculate the probability of a derivation. The Earley parsing algorithm builds trees via a left-most derivation, while the left corner parsing algorithm builds trees via a left-corner derivation. Manning and Carpenter observe that for each parse tree, there is a unique left-corner derivation, and therefore the probability of the complete parse tree must equal the probability of the derivation used to build it. Manning and Carpenter capitalize on this insight to relate the probability of a parse tree and the actions of the parser:

$$\begin{aligned} \Pr(tree) &= \Pr(derivation_{tree}) && (5.2)\\ &= \Pr(op_0 \ldots op_n) && (5.3)\\ &= \Pr(\, op_n \mid op_{n-1}, \ldots, op_0\,) \times \ldots \times \Pr(op_0) && (5.4) \end{aligned}$$

Assuming that a derivation of a tree has n steps, then one can define the probability of a complete parse tree to be the product of all the derivational steps used to build that tree. [8] A partial derivation can admit multiple parse trees. Manning and Carpenter apply a Markov style assumption on equation 5.4, choosing to condition on recent history instead of the complete history.

Given a derivation D, they approximate the probability of each left corner parser operation using the following assumptions:

$$\begin{aligned} \Pr(\, push\ w_i \mid D\,) &\approx \Pr(\, w_i \mid current\ goal\ symbol\,) && (5.5)\\ \Pr(\, attach \mid D\,) &\approx \Pr(\, attach \mid stack\ top\ , goal\ symbol\,) && (5.6)\\ \Pr(\, propose\ rule \mid D\,) &\approx (1 - \Pr(\, attach \mid D\,))\Pr(\, rule \mid stack\ top, goal\,) && (5.7) \end{aligned}$$

[8] Note that the probability of partial derivations does not necessarily equal the probability of a partial tree.

A push can only happen when the top of the stack is incomplete. However if the top of the stack is complete, and the top of the stack is the same as the goal state, there is uncertainty about whether an attach operation or a propose operation should take place. The probability of an attach in equation 5.6 is only nonzero when the goal state and the top of the stack are the same symbol. The probability of a propose a particular rule is shown in equation 5.7. The probability of a propose is the product of one minus the probability of attaching (which is the chance of not attaching) times an estimate of the likelihood of proposing a particular rule given the goal state and the top of the stack.

Manning and Carpenter use their probabilistic left corner parser to parse sentences from the Penn Treebank. The training trees allow them to estimate the probability of each operation directly. Estimating the probabilities in this manner makes sense when given a large set of training data such as the Treebank.

For the construction analyzer's version of left corner parsing, I have extended Manning and Carpenter's approach in the following ways:

- A semantic factor is included to estimate the likelihood of a derivation
- Manning and Carpenter estimate the probability of each operation from data, however there is no ECG treebank. Instead, the probability of each operation can be defined in terms of a probabilistic grammar, and in particular the syntactic terms in the factored syntactic-semantic model defined in the previous chapter.
- In Manning and Carpenter's formulation, they define the joint probability of a left corner derivation with the words. For modeling the reading time data in chapter 7, the conditional probability of each derivation given the words is easier to link to reading time predictions as is done by Hale [30]. Thus I compute the conditional probability of a derivation given the input.

5.3.2 Probabilistic Reachability

The Earley parsing algorithm is a chart-based top-down parser with bottom-up filtering. An Earley parser predicts rules in a top-down fashion, using a single item in the chart to represent all the different ways that a symbol could be predicted. In effect, the chart's single item for a symbol cuts left recursion off at the earliest point, and therefore it is not a problem.

Stolcke [80] observes that special care must be taken to properly accumulate probability mass in the chart. In a naive probabilistic definition of an Earley parser, one would just accumulate the probability of an item i as each new item j predicted it.[9] Left recursion foils this naive implementation, however, because the mass for the left-recursive paths never gets accumulated in the item (the chart cuts off the recursion). Thus Stolcke introduces additional probabilistic machinery to accumulate this lost mass.

Stolcke defines a relation R which is the probabilistic reachability relation allowing for recursion. Intuitively, R encodes the sum of probabilities over all paths between two symbols given the grammar. For PCFGs, Stolcke defines R with the following recurrence relation:

$$\Pr(\, Z \mid X \,) = \sum_{X \to Z\lambda \in G} \Pr(X \to Z\lambda) \tag{5.8}$$

$$R(X,Y) = \delta(X,Y) + \sum_{Z} \Pr(\, Z \mid X \,) R(Z,Y) \tag{5.9}$$

Where $\delta(X,Y) = 1$ if $X = Y$ and $\delta(X,Y) = 0$ if $X \neq Y$. Probability $\Pr(\, Z \mid X \,)$ sums the probabilities of all rules that have X as their left hand side and Z as their immediate left corner. The R relation is straightforward to calculate as the sum of a geometric series or up to a fixed depth using a dynamic programming algorithm as is done in the implemented system. Stolcke observes that the R value can be greater than one especially for reflexive $R(X,X)$ entries.

Probabilistic reachability is an important concept for my derivation of probabilistic left corner parsing because two consecutive items on the stack represent two

[9] This would be the probability of the item j times the probability of the rule used to predict i.

endpoints of a possibly infinite set of parse tree paths. The next section shows how probabilistic reachability is applied to estimate the probability of each parser operation. Additionally, the recurrence relation defined in 5.9 must be modified for ECG. Because ECG allows for productive omission, the notion of a left corner is different than in a PCFG. Section 5.4 defines the necessary extensions to the left corner parsing algorithm for parsing ECG.

5.3.3 A Probabilistic Left Corner Parsing Algorithm

The probabilistic extension of left corner parsing presented here is inspired[10] by inference in Hidden Markov Models (HMM) [71]. Like an HMM, the parser is incrementally given word zero through word i ($w_0 \dots w_i$), and after each word w_i, the parser must infer the most likely stack (or derivation). From this point on, I will use variable t to refer to a left corner derivation instead of a tree, and thus I assume the parser is looking to find

$$argmax_{t_i} \Pr(\, t_i \mid w_{0\dots i}\,)$$

Which is the most likely derivation t_i after seeing word w_i

Expression $\Pr(\, t_i \mid w_{0\dots i}\,)$ is decomposed using Bayes rule, and the resulting equation is known as the forward algorithm (up through equation 5.11): $\Pr(\, t_i \mid w_{0\dots i}\,) =$

$$\begin{aligned}
&= \alpha \Pr(\, w_i \mid t_i, w_{0\dots i-1}\,) \Pr(\, t_i \mid w_{0\dots i-1}\,) && (5.10)\\
&= \alpha \Pr(\, w_i \mid t_i\,) \sum_{t_{i-1}} \Pr(\, t_i \mid t_{i-1}\,) \Pr(\, t_{i-1} \mid w_{0\dots i-1}\,) && (5.11)\\
&= \alpha \Pr(\, w_i \mid t_i\,) \Pr(\, t_i \mid t_{i-1}\,) \Pr(\, t_{i-1} \mid w_{0\dots i-1}\,) && (5.12)
\end{aligned}$$

Equation 5.10 uses Bayes rule to separate w_i from the rest of the words. Line 5.11 applies the standard Markov assumptions and introduces the probability over derivations from the previous iteration, making the equation recursive. In equation 5.12, the summation is dropped because each derivation t_i can only be generated by a single t_{i-1}. The final terms of equation 5.12 are:

[10]The parser itself is not an HMM because the state space is not fixed. However the forward algorithm is an insightful idea for modeling the inference of incremental parsing.

- α: The normalizing term from Bayes rule which is $\Pr(w_0 \ldots w_i)$.
- $\Pr(\, w_i \mid t_i \,)$:The probability of generating the next word given the current analysis. In HMM terminology, this is known as the emission model.
- $\Pr(\, t_i \mid t_{i-1} \,)$: The probability of the analysis that generates w_i given an analysis that generates w_{i-1}. In HMM terminology, this is called the transition model.
- $\Pr(\, t_{i-1} \mid w_{0 \ldots i-1} \,)$: the recursive calculation for t_{i-1};

Unlike hidden Markov models, the t_i's are not simple atomic states, but structured states representing derivations, and often it will take more than one parser operation to turn a derivation that incorporates w_{i-1} into a derivation that incorporates w_i. Thus $\Pr(\, t_i \mid t_{i-1} \,)$ is actually a shorthand for the likelihood of a sequence of parser operations. Assuming there are $j+1$ steps to extend parse t_{i-1} to parse t_i:

$$\Pr(\, t_i \mid t_{i-1} \,) = \Pr(\, t_i \mid t_{i,j-1} \,) \ldots \Pr(\, t_{i,1} \mid t_{i,0} \,) \Pr(\, t_{i,0} \mid t_{i-1} \,)$$

This decomposition assumes the standard Markov assumption that the successor interim parser state only depends on the immediate prior state.

However, decomposing $\Pr(\, t_i \mid t_{i-1} \,)$ in this manner raises a question about finding "good" interim parser states between t_{i-1} and t_i. This subproblem can also be cast as a search task with the start state being a parser state that incorporates word $i-1$ and a descendant "goal" parser state that incorporates word w_i. The system uses the probability of the next word given an interim state or $\Pr(\, w_i \mid t_{i,k} \,)$ as a search heuristic. Details regarding how $\Pr(\, w_i \mid t_{i,k} \,)$ is calculated are provided in section 5.3.6.

5.3.4 The Likelihood of each Parser Operation

Upon comparing the actions of the left corner parser with the HMM model, one notices that the push action is similar to the HMM's emission action, suggesting that a push action could be probabilistically modeled as the emission probability $\Pr(\, w_i \mid t_i \,)$ where t_i is a stack. As a simplifying (Markov) assumption, I assume

that the likelihood of the w_i only depends on the top of the stack. Using the stack notation from section 5.2: $\Pr(\, w_i \mid t_i \,) = \Pr(\, w_i \mid \alpha_j[\beta_j, \gamma_j] \,)$ where α_j is the top of t_i. In particular, β_j must be the constituent that (indirectly) generates w_i, so $\Pr(\, w_i \mid t_i \,) = \Pr(\, w_i \mid \beta_j \,)$.

To define $\Pr(\, w_i \mid \beta_j \,)$, I rely on the probabilistic reachability relation R . Normalizing R over words reachable from a constituent provides the basis for an estimate of $\Pr(\, w_i \mid \beta_j \,)$:

$$\Pr(\, w_i \mid \alpha_j[\beta_j, \gamma_j] \,) = \Pr(\, w_i \mid \beta_j \,) = \frac{R(\beta_j, w_i)}{\sum_{w \in words} R(\beta_j, w)} \tag{5.13}$$

In equation 5.13, $R(\beta_j, w_i)$ is normalized over the set of lexemes in the grammar. This provides an estimate of how often w_i is pushed compared to other lexical items in the grammar, and it is straightforward to see that it sums to one over all lexical items reachable from β_j.

As for the "transition model", assuming there are $k+1$ steps to extend derivation t_{i-1} to derivation t_i then:

$$\Pr(\, t_i \mid t_{i-1} \,) = \Pr(\, t_{i,k} \mid t_{i,k-1} \,) \ldots \Pr(\, t_{i,1} \mid t_{i,0} \,) \Pr(\, t_{i,0} \mid t_{i-1} \,)$$

Propose and attach actions are the "transition" actions between interim derivations $t_{i,x}$ and $t_{i,x+1}$ where $0 < x < k$. Like Manning and Carpenter (in equation 5.4), I assume that the conditional probability of the next interim derivation state is the probability of the parser action that generated that state:

$$\Pr(\, t_{i,x+1} \mid t_{i,x} \,) = \Pr(\, op_x \mid t_{i,x} \,)$$

Then like Manning and Carpenter, I summarize the information in derivation $t_{i,x}$ with the goal state and the current left corner. More precisely, if

$$t_{i,x} = \alpha_0[\beta_0\gamma_0], \ldots \alpha_{i-1}[\beta_{i-1}\gamma_{i-1}], \alpha_i[]$$

then I assume:

$$\Pr(\, t_{i,y} \mid t_{i,x} \,) \approx \Pr(\, t_{i,y} \mid \alpha_i, \beta_{i-1} \,) = \Pr(\, op \mid \alpha_i, \beta_{i-1} \,)$$

With this simplification, the probability of each operation is defined as:

$$\Pr(\, attach \mid \alpha_i, \beta_{i-1}\,) = \frac{\Pr_a(\, \alpha_i \mid \beta_{i-1}\,)}{R(\beta_{i-1}, \alpha_i)} \tag{5.14}$$

$$\Pr(\, propose\ \alpha' \rightarrow \alpha_i\lambda \mid \alpha_i, \beta_{i-1}\,) = \frac{\Pr(\alpha' \rightarrow \alpha_i\lambda) R(\beta_{i-1}, \alpha')}{R(\beta_{i-1}, \alpha_i)} \tag{5.15}$$

The numerator of equation 5.14 is the probability[11] of attaching α_i to β_{i-1}. For a PCFG, $\Pr_a(\, \alpha_i \mid \beta_{i-1}\,)$ is one if $\alpha_i = \beta_{i-1}$ and zero otherwise. In the numerator of equation 5.15, $\Pr(\alpha' \rightarrow \alpha_i\lambda)$ accounts for the probability of the rule $\alpha' \rightarrow \alpha_i\lambda$ being used. In the PCFG setting, this is the conditional probability of the rule given α'. The term $R(\beta_{i-1}, \alpha')$ accounts for the probability of the sum of all the paths between β_{i-1} and α'. The product of $\Pr(\alpha' \rightarrow \beta'...)$ and $R(\beta_{i-1}, \alpha')$ provides the sum of the probabilities for all paths between β_{i-1} and α_i that attach to α_i using rule $\alpha' \rightarrow \alpha_i\lambda$.

It is straightforward to show that the appropriate normalizer for the attach and propose operations is $R(\beta_{i-1}, \alpha_i)$. The numerators for the attach and propose operations implement the recurrence relation definition of R given in equation 5.9:

$$R(\beta_{i-1}, \alpha_i) = \Pr_a(\, \alpha_i \mid \beta_{i-1}\,) + \sum_{\rho \in \alpha' \rightarrow \alpha_i\lambda} \Pr(\rho) R(\beta_{i-1}, \alpha')$$

Where $\Pr_a$ corresponds to the single-step δ function from equation 5.9 and all the rules that can be proposed are summed. Thus the conditional probability function $\Pr(\, op \mid a_{i,x}\,)$ integrates to one.

5.3.5 An Example

For a complete example of the algorithm at work, consider the following probabilistic context free grammar G and associated R values:[12]

[11] $\Pr_a$ is being used here to stress that this is the attach probability and not the normalized R probability defined in equation 5.13

[12] Entries not shown in the table are assumed to be zero

Rule	**Probability**
S → A	1
A → a	1/3
A → Aa	1/3
A → AB	1/3
B → bB	1/2
B → b	1/2

Constituent(*type*)	*Symbol*	*R*
A	A	3
A	a	1
a	a	1
b	b	1
B	b	1
B	B	1

In grammar G, lowercase letters are terminals and upper case letters are nonterminals. Additionally, the R values are computed using the formula for the sum of a geometric series.[13]

The grammar G recognizes strings of combined "a"s and "b"s, starting with an "a". For the string "aabb", the operations that the parsing algorithm performs are shown in figure 5.2. Figure 5.2 also shows the conditional probability of the stack resulting from the parser operation. For example, line three specifies that the initial stack is of the form $S[A], A[]$ and that a propose operation with rule A → Aa is necessary. The conditional probability of the result stack $S[A], A[a]$ is 1/3. With the initial stack $S[A], A[]$, two other parser operations are possible. Item $A[]$ could be attached to $S[A]$, or the rule A → AB could be used instead. Both of those operations also have a conditional probability of 1/3.

5.3.6 Search

Thinking of parsing in terms of search, the left corner parsing algorithm specifies the operators for extending search states (in-progress derivations) and computing their probabilities, but with the ambiguity of natural language, more work needs to

[13]The formula is $a/(1-r)$ with a being the start term and r being the common ratio (when r is less than one).

	Stack	**Parser Op**	**Conditional Stack Prob**
1	$S[A]$	Push(a)	$\Pr(\, a \mid A \,) = 1$
2	$S[A], a[]$	Propose(A $\rightarrow$ a)	$\Pr(A \rightarrow a) R(A, A)/R(A, a) = 1$
3	$S[A], A[]$	Propose(A $\rightarrow$ Aa)	$\Pr(A \rightarrow Aa) R(A, A)/R(A, A) = 1/3$
4	$S[A], A[a]$	Push(a)	$\Pr(\, a \mid a \,) = 1$
5	$S[A], A[a], a[]$	Attach	$\Pr_a(\, a \mid a \,)/R(a, a) = 1$
6	$S[A], A[]$	Propose(A $\rightarrow$ AB)	$\Pr(A \rightarrow AB) R(A, A)/R(A, A) = 1/3$
7	$S[A], A[B]$	Push(b)	$\Pr(\, b \mid B \,) = 1$
8	$S[A], A[B], b[]$	Propose(B $\rightarrow$ bB)	$\Pr(B \rightarrow bB) R(B, B)/R(B, b) = 1/2$
9	$S[A], A[B], B[B]$	Push(b)	$\Pr(\, b \mid B \,)$
10	$S[A], A[B], B[B], b[]$	Propose(B $\rightarrow$ b)	$\Pr(B \rightarrow b) R(B, B)/R(B, b) = 1/2$
11	$S[A], A[B], B[B], B[]$	Attach	$\Pr_a(\, B \mid B \,)/R(B, B) = 1$
12	$S[A], A[B], B[]$	Attach	$\Pr_a(\, B \mid B \,)/R(B, B) = 1$
13	$S[A], A[]$	Attach	$\Pr_a(\, A \mid A \,)/R(A, A) = 1/3$

Figure 5.2: The operations that the left corner parsing algorithm executes for the single correct derivation of the string "aabb". The first column shows the stack prior to the operation shown on the same line in the "Parser Op" column. Column "Conditional Stack Prob" shows the probability of the action given just the initial stack. The stack that results from the action is shown on the subsequent line. For example, on line one, the initial stack is $S[A]$, the parser action is to push "a", and the conditional likelihood of the push action given $S[A]$ is one.

be done to improve the efficiency of the search. Many broad-coverage parsers sacrifice completeness for performance using various beaming techniques [11] [65]. Other parsers opt for agenda-based parsing, but use heuristic techniques for estimating cost to completion of a parse. The figures of merit work [6] used in the Charniak parser [10] offers high performance parsing without the theoretical guarantee of completeness, while broad coverage A^* parsers such as Klein and Manning's [39] and Haghighi, DeNero and Klein [27] guide the search with provably optimistic heuristics that do guarantee completeness. Petrov and Klein [62] apply a novel technique to parsing by iteratively performing a complete parse of a sentence with a simpler grammar, then using those results to constrain a complete parse with a more sophisticated grammar. In essence, the goal states from a simpler search space constrain the search space on the next iteration. Their scheme is both fast and effective, yielding state of the art performance.

For my constructional analyzer, the search algorithm is limited to techniques that are cognitively and psychologically plausible. Complete parsing strategies are questionable because of memory limitation requirements, and cost-to-completion estimates are limited to heuristics that can be computed incrementally without lookahead. Given these constraints, two search algorithms have been implemented:

- A standard beam search algorithm that iteratively extends derivations from those that incorporate word w_i to those that incorporate word w_{i+1}. Two priority queues are used to implement this solution, and each priority queue is limited to a finite, user-specified size. With this search strategy, the surviving derivations proceed in a near lock step together. This is the parser that is used for the psycholinguistic results.

- A limited-memory agenda-based parser that uses a single priority queue. In principle, this setup allows for a single highly likely parse to proceed to the end of the sentence without waiting for other derivations to catch up. Additionally, for applied information extraction tasks and grammar building, this search technique is easier to use because one does not have to worry as much about derivations falling off the beam.

Given that a derivation has incorporated word w_i, the parser should perform push, attach and propose actions that result in states that are likely to generate word w_{i+1}. This intuition is consistent with Roark who uses the exact same heuristic to speed up his parser [68], and Henderson who adds the next word to his neural net that conditions what parser action to do next [32]. Thus both the beam and agenda-based parsing strategies use $\Pr(\, w_{i+1} \mid \mathit{derivation}\,)$ as a psychologically plausible and extremely optimistic estimate of the cost to complete a derivation. If the top item of the derivation stack is incomplete and has remaining goal constituents, the probability of word w_{i+1} is the same as the probability of pushing word w_{i+1}. If the top of the stack is complete, however, computing $\Pr(\, w_{i+1} \mid \mathit{derivation}\,)$ is much more expensive. Before I show how to calculate $\Pr(\, w_{i+1} \mid \mathit{derivation}\,)$, some examples will be helpful for developing the intuition.

Assume that stack ψ is of the form $\alpha_0[\beta_0\gamma_0], \ldots, \alpha_{k-1}[\beta_{k-1}\gamma_{k-1}], \alpha_k[]$. Then w_{i+1} could be predicted by constituent γ_{k-1}, but only if α_k attached directly to β_{k-1} or through a path of unary productions. Additionally, another rule λ could be proposed in between α_k and α_{k-1}. This hypothetical item $\lambda[\beta_{\lambda,0}, \beta_{\lambda,1}]$ could attach α_k to $\beta_{\lambda,0}$ (or through a path of unary productions), and then constituent $\beta_{\lambda,1}$ could predict the next word w_{i+1}.

Now consider a stack of the form $\alpha_0[\beta_0\gamma_0], \ldots, \alpha_{k-2}[\beta_{k-2}\gamma_{k-2}], \alpha_{k-1}[\beta_{k-1}], \alpha_k[]$. Either via direct attachment or the use of a chain of unary productions from β_{k-1} to α_k, item α_{k-1} can be completed. As a consequence, constituent γ_{k-2} would then become capable of predicting word w_{i+1}. The probability of word w_{i+1} given γ_{k-2} would have to be scaled by the probability of completing item α_{k-1}.

Generally speaking, an item on the stack can be used to generate the next word if everything above it can be completed with just attaches or unary productions. Using this intuition (and ignoring semantics), one can define a formula for the probability of word w_{i+1} given a stack ψ. Assuming ψ has ψ_k items on it, one can approximate $\Pr(\, w_{i+1} \mid \psi\,)$ in the following way:[14]

$$\Pr(\, w_{i+1} \mid \psi\,) \;\approx\; \sum_{\kappa \in \psi_k \ldots 1} \delta(\psi, \kappa)(P_{att(\psi,\kappa,w_{i+1})} + P_{prop(\psi,\kappa,w_{i+1})}) \tag{5.16}$$

[14] R_u is the reachability using only unary rules.

$$\delta(\psi,\kappa) = \prod_{\mu\in\psi_k-1\ldots\kappa} if \mid \psi.\alpha_\mu.goals \mid = 1 \; then \frac{R_u(\beta_\mu,\alpha_{\mu+1})}{R(\beta_m u,\alpha_{\mu+1})} \; else \; 0 \quad (5.17)$$

$$P_{att(\psi,\kappa,w_{i+1})} = \frac{R_u(\beta_{\kappa-1,0},\alpha_\kappa)\Pr(\, w_{i+1} \mid \beta_{\kappa-1,1}\,)}{R(\beta_{\kappa-1,0},\alpha_\kappa)} \quad (5.18)$$

$$P_{prop(\psi,\kappa,w_{i+1})} = \sum_{\lambda\in G} \frac{R_u(\beta_{\lambda,0},\alpha_\kappa)R(\beta_{\kappa-1,0},\lambda)\Pr(\, w_{i+1} \mid \beta_{\lambda,1}\,)}{R(\beta_{\kappa-1,0},\alpha_\kappa)} \quad (5.19)$$

Summing over each $item_\kappa$ on the stack, equation 5.16 computes the likelihood of $item_\kappa$ completing and $item_{\kappa-1}$ playing some role in generating the next word. The δ function computes the likelihood of everything through and including $item_\kappa$ completing. P_{att} computes the likelihood that $item_\kappa$ is attached directly to $item_{\kappa-1}$'s constituent $\beta_{\kappa-1,0}$, thereby allowing $\beta_{\kappa-1,1}$ to play a role in generating the next word. P_{prop} computes the likelihood that some intermediate item is proposed between $item_\kappa$ and $item_{\kappa-1}$, and it is this intermediate item that plays a role in generating the next word.

In equation 5.18, the probability of attaching (through a direct attach or unary productions $item_\kappa$ to to first unmatched constituent of $item + \kappa - 1$ is calculated by normalizing the unary reachability R_u by the total reachability between $item_{\kappa-1}$ and $item_\kappa$. The normalized unary reachability then acts as a scaling factor on the probability of generating the next word with $\beta_{\kappa-1,1}$.

Equation 5.19 sums over the different rules (λ) that could be proposed between $item_\kappa$ and $item_{\kappa-1}$. The likelihood of rule λ in this scenario is the unary reachability between λ's first constituent $\beta_{\lambda,0}$ and α_κ times the reachability between $item_{\kappa-1}$ and λ normalized by the total reachability between $item_{\kappa-1}$ and $item_\kappa$. This probability scales the likelihood of the next unmatched constituent of λ generating the next word.

For the PCFG case, the δ function in equation 5.17 is easy to define. An item on the stack can only complete if everything above it on the stack completes, and it has a single unmatched constituent. Equation 5.17 encodes constrains the stack item μ to have just a single remaining constituent (or goal) with the if clause, and if there is just a single remaining constituent, the likelihood of completing is the normalized unary reachability between the item and item just above it on the stack. Otherwise the item cannot possibly be completed, and the δ function returns zero.

The δ function plays a second role at the end of the sentence. When a derivation

has incorporated all the words of the input, there are no more words to generate, and obviously Pr(*next word* | *derivation*) is not a useful heuristic for guiding the parser. In this case, what is important is the cost of completing the derivation (i.e. completing every item on its stack), and computing the δ function for the whole stack provides an estimate of the completion likelihood.

5.4 Extending the Left Corner Parsing Algorithm for ECG

Section 5.3 derived a probabilistic left corner parsing algorithm for PCFGs. ECG is a more expressive grammar formalism, requiring extension to both the algorithm and the probabilistic derivation. Productive argument omission, for example, leads to an extended definition for what can be an expected (goal) symbol for a given item on the stack. Additionally, the equations for approximating the probability of each parser operation must be rewritten in terms of the factored syntactic and semantic model from the previous chapter.

5.4.1 Constructional Factor Summary

Before rewriting the left corner probability equations in terms of the factored syntactic and semantic model, it will he helpful to review the parameters used in the syntactic factor. Assuming construction α, constituent β and filler type θ, the probabilistic syntactic parameters employed in the model are:

- Pr($expressed_\beta$ *vs.* $omitted_\beta$ | α): This is the probability that constituent β is expressed or not expressed (omitted) given the construction. Note that this term does not depend on the context, as it models a construction's preference for omitting an argument ignoring context, letting the reference resolution incorporate context.

- Pr($local_\beta$ *vs.* $nonlocal_\beta$ | $\alpha, expressed_\beta$): The probability of construction α's constituent β being expressed locally (as a normal constituent) or expressed

nonlocally as a fronted constituent

- $\Pr(\, filler = \theta \mid \alpha, expressed_\beta \,)$: Assuming β is expressed, this term is the likelihood that a particular constructional type θ fills constituent β.

To shorten the equations in the following sections, I abbreviate $expressed_\beta$ with exp_β, $omitted_\beta$ with exp_β $local_\beta$ with loc_β and $\Pr(\, filler = \lambda \mid \alpha, expressed_\beta \,)$ with $\Pr(\, \lambda \mid \alpha, exp_\beta \,)$.

5.4.2 Updating Probabilistic Reachability

ECG allows for productive argument omission. This means that for a given a stack item $\alpha[\beta\gamma\lambda]$, β, γ, and λ could each be the expected (goal) symbol. Additionally, given that the "right hand side" of an ECG rule is not required to be a total ordering, the notion of which constituent comes next for a given stack item must be refined. Instead of a single symbol β being expected or "next", there must be a set of next symbols.

It is straightforward to statically infer which constituents of a construction α must be matched before constituent β. This information is stored in a precedes relation $precedes(\beta)$. Doing this takes into account the partial constituent ordering that ECG allows. During parsing, each stack item α_i maintains a set of unmatched constituents $unmatched(\alpha_i)$ and a set of next constituents $next(\alpha_i)$. If none of the constituents in $precedes(\beta_{i,j})$ are in $unmatched(\alpha_i)$, then $\beta_{i,j}$ is in $next(\alpha_i)$.

Given these definitions and the terms of the constructional factor summarized in section 5.4.1, the static definition of probabilistic reachability defined in equation 5.9 can be updated. For construction α with constituent β and possible filler λ, the R relation is redefined in the following way:

$$
\begin{aligned}
R(\alpha.\beta, \lambda) &= \Pr(\, \lambda \mid \alpha, exp_\beta \,) && (5.20)\\
&+ \sum_{\rho \in G} R(\alpha.\beta, \rho) \sum_{\gamma \in \rho} P_{loc(\rho,\gamma)} \Pr(\, \lambda \mid \rho, exp_\gamma \,) P_{omit(\rho,\gamma)} && (5.21)\\
P_{loc(\rho,\gamma)} &= \Pr(\, exp_\gamma \mid \rho \,) \Pr(\, loc_\gamma \mid \rho, exp_\gamma \,) && (5.22)\\
P_{omit(\rho,\gamma)} &= \prod_{\theta \in precedes(\gamma)} \Pr(\, omit_\theta \mid \rho \,) && (5.23)
\end{aligned}
$$

Instead of making the R a relation from symbol to symbol, it is now a relation from a particular constructional constituent $\alpha.\beta$ to a construction type λ. Line 5.20 is the single step attachment cost for type λ to constituent $\alpha.\beta$. Line 5.21 is the recursive part of the R relation definition which computes the reachability between β and λ through each construction ρ in grammar G. For a particular construction ρ, one multiplies the R value for β and ρ by the chance that construction λ can attach to some constituent γ in ρ (i.e. summed over all γ). To compute the probability that λ can be attached to constituent γ in ρ, one multiplies the probability that γ is expressed locally (equation 5.22) by the probability of filling ρ with λ times the probability of omitting all of the constituents in $precedes(\gamma)$ (equation 5.23). Omitting each of the constituents in $precedes(\gamma)$ is necessary to allow γ to be matchable.

Note that calculating the R relation in this way does not take into account nonlocal constituents, and thus it sacrifices correctness for efficiency. Computing the R relation is expensive, and the factors that condition nonlocal constituents are obviously not available prior to parse time.[15]

During parsing, an additional dynamic definition of probabilistic reachability must also be calculated between items on a stack. This online version of the R relation (called R_o) takes into account which constituents have been matched and also makes use of the expressed but nonlocal constituents. Given a stack item of the form $\ldots \alpha_j[\beta_{j,0}, ..., \beta_{j,k}], \alpha_{j+1}$, R_o can be calculated in the following way:

$$R_o(\alpha_j[\beta_{j,0}, .., \beta_{j,k}], \alpha_{j+1}) = \sum_{\beta \in \beta_j} (P_{omit(\alpha_j,\beta)} + P_{nonloc(\alpha_j,\beta)}) P_{loc(\alpha_j,\beta)} R(\beta, \alpha_{j+1}) \quad (5.24)$$

$$P_{loc(\alpha_j,\beta)} = \Pr(\, exp_\beta \mid \alpha_j \,) \Pr(\, loc_\beta \mid \alpha_j, exp_\beta \,) \quad (5.25)$$

$$\Upsilon_{\alpha_j,\beta} = precedes(\beta) \cap unmatched(\alpha_j) \quad (5.26)$$

$$P_{omit(\alpha_j,\beta)} = \prod_{c \in \Upsilon_{\alpha_j,\beta}} \Pr(\, omit_c \mid \alpha_j \,) \quad (5.27)$$

$$P_{nonloc(\alpha_j,\beta,\lambda)} = \sum_{c \in \Upsilon_{\alpha_j,\beta}} P_{nonlocfiller(\alpha_j,c,\lambda)} \prod_{d \in \Upsilon_{\alpha_j,\beta} \setminus c} \Pr(\, omit_d \mid \alpha_j \,) \quad (5.28)$$

[15] There is an unimplemented extension of reachability that corrects this problem. Instead of using the predefined R value directly, one could try every construction η that was reachable from β and can generate λ. The constituents of η would be allowed to be nonlocal, and the reachability between β and λ would be re-computed in terms of η.

$$P_{nonlocfiller(\alpha_j,\beta,\lambda)} = \Pr(\, exp_\beta \mid \alpha_j \,) \Pr(\, loc_\beta \mid \alpha_j, exp_\beta \,) \times \Pr(\, \lambda \mid \alpha_j, \exp_\beta \,)$$

Equation 5.24 defines R_o. R_o takes a stack item $\alpha_j[\beta_{j,0}, ..., \beta_{j,k}]$ and a type α_{j+1} as arguments. It calculates the total mass between the item α_j and α_{j+1} by summing over the R values for each remaining unmatched constituent in α_j. For each remaining constituent β in $\alpha_j[\beta_{j,0}, ..., \beta_{j,k}]$, a weighted $R(\beta, \alpha_{j+1})$ is summed in. The weights are $P_{omit(\alpha_j,\beta)}$ and $P_{nonloc(\alpha_j,\beta)}$.

Equation 5.26 defines a set Υ that represents the unmatched constituents of stack item α_j that are also in the precedes set of β. These constituents must be either be omitted or treated as nonlocal before β can be employed in the derivation. Equation 5.27 calculates the probability of omitting all of the constituents in Υ so that β can be used to generate α_{j+1}. $P_{nonloc(\alpha_j,\beta)}$ in equation 5.28 is the probability that exactly one of the constituents (constituent c) in Υ is matched by a fronted constituent with the rest being omitted. $P_{nonlocfiller(\alpha_j,\beta,\lambda)}$ calculates the likelihood that the fronted constituent λ is bound to constituent c by factoring in the probability that c is expressed, nonlocal, and filled by λ.

5.4.3 Modifying the Parser Operations for ECG

The intuition behind the equations for calculating the probability of each parser operation is the same for ECG as it is for the equations for PCFGs defined in section 5.3.3. The conditional probability $\Pr(\, t_{i,j} \mid t_{i-1,j-1} \,)$ still equals $\Pr(\, op_j \mid t_{i-1,j} \,)$. and the terms used in the equations have similar meaning and are normalized in the same way.

When the Top of the Stack is Complete

Just as before, when the top of the stack is complete, only attach and propose operations can be performed on a derivation. To define the probability of an attach operation, assume a stack ψ of the form:

$$\alpha_0[\beta_{0,0}, ...\beta_{0,k_0}], \ldots \alpha_{i-1}[\beta_{i-1,0}, ..., \beta_{i-1,k_{i-1}}], \alpha_i[]$$

in which α_i is attached to β where $\beta \in \beta_{i-1,0}, ..., \beta_{i-1,k_{i-1}}$:
$\Pr(\, attach\ \beta \mid \alpha_{i-1}[\beta_{i-1,0}, ..., \beta_{i-1,k_{i-1}}], \alpha_i[]\,)$

$$= \frac{(P_{omit(\alpha_{i-1},\beta)} + P_{nonloc(\alpha_{i-1},\beta)})P_{fill(\alpha_{i-1},\beta,\alpha_i)}}{R_o(\alpha_{i-1}[\beta_{i-1,0}, ..., \beta_{i-1,k_{i-1}}], \alpha_i[])} \tag{5.29}$$

where:

$$\begin{aligned}
P_{fill(\alpha i-1,\beta,\alpha_i)} &= \Pr(\, exp_\beta \mid \alpha_{i-1}\,) \Pr(\, loc_\beta \mid \alpha_{i-1}, exp_\beta\,) \times \\
&\quad \Pr(\, \alpha_i \mid \alpha_{i-1}, exp_\beta\,) && (5.30)\\
\Upsilon_{\alpha_{i-1},\beta} &= precedes(\beta) \cap unmatched(\alpha_{i-1}) && (5.31)\\
P_{omit(\alpha_{i-1},\beta)} &= \prod_{c \in \Upsilon_{\alpha_{i-1},\beta}} \Pr(\, omit_c \mid \alpha_{i-1}\,) && (5.32)\\
P_{nonloc(\alpha_{i-1},\beta,\lambda)} &= \sum_{c \in \Upsilon_{\alpha_{i-1},\beta}} P_{nonlocfiller(\alpha_{i-1},c,\lambda)} \prod_{d \in \Upsilon_{\alpha_{i-1},\beta} \setminus c} \Pr(omit_d \mid \alpha_{i-1}) && (5.33)\\
P_{nonlocfiller(\alpha_{i-1},c,\lambda)} &= \Pr(\, exp_c \mid \alpha_{i-1}\,) \Pr(\, nonloc_c \mid \alpha_{i-1}, exp_c\,) \times \\
&\quad \Pr(\, \lambda \mid \alpha_{i-1}, \exp_c\,) && (5.34)
\end{aligned}$$

Like the PCFG case, equation 5.29 is normalized by the reachability between α_{i-1} and α_i. And like the PCFG case, the numerator is the generative probability of making α_i the filler of constituent $\alpha_{i-1}.\beta$. P_{fill} (equation 5.30) calculates the likelihood of constituent β being expressed locally and filled by α_i. However, attaching α_i to constituent β assumes that all of the unmatched constituents in $precedes(\beta)$ can be omitted or treated as nonlocal. Term P_{omit}, defined in equation 5.32, computes the likelihood of all the unmatched constituents in $precedes(\beta)$ being omitted. Term P_nonloc, defined in equation 5.33, calculates the probability of exactly one of the constituents in $precedes(\beta)$ being expressed nonlocally and filled by the fronted item λ (equation 5.34), and the rest of the unmatched constituents being omitted. P_{omit} and P_{nonloc} are added together because either scenario would enable α_i to attach to β.

To make this more concrete, consider the example sentence, *the quarterback passes to the receiver for the score* which might be uttered by the announcer during a football game. Assuming that the correct analysis of this sentence employs a cause-motion construction with an omitted theme, the derivation of this sentence after completing

the PP would look like:

$$Declarative[VP], CauseMotion[theme, path], PP[]$$

The probability of attaching the PP item to the *path* constituent of the CauseMotion incorporate the following terms:

$$\begin{aligned} P_{fill} &= \Pr(\, exp_{path} \mid CM \,) \Pr(\, loc_{path} \mid CM \,) \Pr(\, PP \mid CM, exp_{path} \,) \\ P_{omit} &= \Pr(\, omit_{theme} \mid CM \,) \\ P_{nonloc} &= 0 \end{aligned}$$

Where CM is an abbreviation for CauseMotion and P_{nonloc} is zero because there are no fronted items available.

While the probability of an attach operation to constituent β is the sum over all different ways of dealing with the unmatched constituents in $precedes(\beta)$, technically the parser must treat the omission and nonlocal assignments as different parser states because they have different semantic ramifications. For example, the omitted constituents must be tied to context, while the fronted constituent's meaning is bound in to the appropriate roles in the semspec.

To define the probability of a propose operation, assume a stack of the form:

$$\alpha_0[\beta_{0,0}, ..., \beta_{0,k_0}], \ldots \alpha_{i-1}[\beta_{i-1,0}, ..., \beta_{i-1,k_{i-1}}], \alpha_i[]$$

and a grammar rule λ of the form $\alpha_\lambda \rightarrow \beta_{\lambda,0}, ..., \beta_{\lambda,k}$ where α_i attaches to β where $\beta \in \beta_{\lambda,0}, ..., \beta_{\lambda,k}$, the likelihood of proposing rule λ is:

$$\begin{aligned} &\Pr(\, propose\ \lambda,\ attach\ \beta \mid \alpha_{i-1}[\beta_{i-1,0}, ..., \beta_{i-1,k_{i-1}}], \alpha_i[] \,) = \\ &\frac{(P_{omit(\lambda,\beta)} + P_{nonloc(\lambda,\beta)}) P_{fill(\lambda,\beta,\alpha_i)} R_o(\alpha_{i-1}[\beta_{i-1,0}, ..., \beta_{i-1,k_{i-1}}], \lambda)}{R_o(\alpha_{i-1}[\beta_{i-1,0}, ..., \beta_{i-1,k_{i-1}}], \alpha_i[])} \end{aligned} \tag{5.35}$$

where:

$$P_{fill(\lambda,\beta,\alpha_i)} = \Pr(\, exp_\beta \mid \lambda \,) \Pr(\, loc_\beta \mid \lambda, exp_\beta \,) \Pr(\, \alpha_i \mid \lambda, exp_\beta \,) \tag{5.36}$$

$$\Upsilon_{\lambda,\beta} = precedes(\beta) \tag{5.37}$$

$$P_{omit(\lambda,\beta)} = \prod_{c \in \Upsilon_{\lambda,\beta}} \Pr(\, omit_c \mid \lambda \,) \quad (5.38)$$

$$P_{nonloc(\lambda,\beta,\lambda)} = \sum_{c \in \Upsilon_{\lambda,\beta}} P_{nonlocfiller(\lambda,c,\lambda)} \prod_{d \in \Upsilon_{\lambda,\beta} \setminus c} \Pr(\, omit_d \mid \lambda \,) \quad (5.39)$$

$$P_{nonlocfiller(\lambda,c,\lambda)} = \Pr(\, exp_c \mid \lambda \,) \Pr(\, nonloc_c \mid \lambda, exp_c \,) \Pr(\, \lambda \mid \lambda, \exp_c \,) \quad (5.40)$$

Equation 5.35 has many of the same terms as equation 5.29. For example, constituent β still needs to be expressed locally and filled by α_i which leads to the similarity between equations 5.36 and 5.30. The constituents in $precedes(\beta)$ must either all be omitted or all but one of them must be omitted with the other constituent in $precedes(\beta)$ set to nonlocal. This is also quite similar to the attach case, and motivates the similarities between equations 5.32, 5.33, 5.34 and equations 5.38, 5.39, 5.40, respectively. The only difference between the formula for the probability of an attach and the formula for the probability of a propose is the inclusion of $R_o(\alpha_{i-1}[\beta_{i-1,0}, ..., \beta_{i-1,k_{i-1}}], \lambda)$ in the numerator which incorporates the reachability between α_{i-1} and the newly proposed symbol λ.

When the Top of the Stack is Incomplete

One major difference between the PCFG case and the ECG case is when the top of the stack is incomplete. With PCFGs, the parser is deterministically required to perform a push operation. However, with ECG's productive omission, there is uncertainty over whether the next word should be pushed onto the stack or whether all of the unmatched constituents on the top of the stack should be omitted (thereby completing the top of the stack). Thus a new parser operation must be added to "finish" an incomplete top item on the stack.

To define the probability of a finish operation, assume a stack ψ in which the top of the stack is incomplete:

$$\alpha_0[\beta_{0,0}, ...\beta_{0,k_0}], \ldots \alpha_i[\beta_{i,0}, ..., \beta_{i,k_i}]$$

then the probability of the finish operation is the same as the probability of omitting

all of the unmatched constituents:[16]

$$\Pr(\, finish \mid \psi\,) = \prod_{\beta \in \beta_{i,0} \ldots \beta_{i,k_i}} \Pr(\, omit_{\beta} \mid \alpha_i\,) \tag{5.41}$$

Given the probability $\Pr(\, finish \mid derivation\,)$, the probability of a push operation can be defined. Assume the top of the stack ψ looks like $\alpha_i[\beta_{i,0}, ..., \beta_{i,k}]$ and the word to be pushed is w_{i+1}. Then the conditional probability of a pushing w_{i+1} given ψ is:

$$\Pr(\, push \mid \psi\,) = (1 - \Pr(\, finish \mid \psi\,))\Pr(\, w_{i+1} \mid \psi\,) \tag{5.42}$$

$$\Pr(\, w_{i+1} \mid \psi\,) = \frac{R_o(\alpha_i[\beta_{i,0}, ..., \beta_{i,k}], w_{i+1})}{\sum_{\beta \in \beta_{i,0},\ldots,\beta_{i,k}} (P_{omit(\alpha_j,\beta)} + P_{nonloc(\alpha_j,\beta)}) P_{loc(\alpha_j,\beta)} \times \sum_{w \in words} R(\beta, w)} \tag{5.43}$$

Equation 5.42 scales the probability of the push operation with (1-$\Pr(\, finish \mid \psi\,)$) to properly model the uncertainty over pushing or finishing given a stack. Apart from that, equation 5.42 is quite similar to equation 5.13. The probability of pushing word w_{i+1} given ψ is the normalized reachability value of that word given the top of the stack.

5.4.4 The Probability of the Next Word

With the introduction of productive omission, it becomes all the more important to guide the search by selecting states that are likely to be on the path to the next word. Updating the estimate of the probability of the next word for ECG with productive omission is not much more complicated given what has already been described. The probability of the next word can still be described with:

$$\Pr(\, w_{i+1} \mid \psi\,) \approx \sum_{\kappa \in \psi_k \ldots 1} \delta(\psi, \kappa)(P_{att(\psi,\kappa,w_{i+1})} + P_{prop(\psi,\kappa,w_{i+1})}) \tag{5.44}$$

To summarize, equation 5.44 approximates the probability of the next word by summing over the likelihood that each item $\alpha_{\kappa-1}$ on the stack predicts the next word. The probability that item $\alpha_{\kappa-1}$ attaches to its descendant α_{κ} and then predicts the

[16] If a fronted constituent is available, then another finish operation that binds one of the unmatched constituents to the fronted constituent is also a possibility.

next word is $P_{att(\psi,\kappa,w_{i+1})}$. The probability that a rule is proposed between item $\alpha_{\kappa-1}$ and α_κ, and that the rule predicts the next word is $P_{prop(\psi,\kappa,w_{i+1})}$). The chance that item $\alpha_{\kappa-1}$ is even capable of predicting the next word is defined by $\delta(\psi,\kappa)$ which computes the likelihood that every item above $\alpha_{\kappa-1}$ on the stack can be finished through some combination of omission, nonlocal constituency and constituency via unary productions.

5.5 Incorporating the Semantic Factor

Up to this point, I have focused on computing the probability of a derivation, ignoring the semantic factor altogether. Computing the probability of derivation τ given the grammar G and words w only takes us part of the way towards computing the probability of an analysis a:

$\Pr(\, a \mid G, w\,) \approx$

$$\Pr(\, assignments \mid fillers, csets\,) \Pr(\, fillers \mid csets, \tau, G, Z\,) \Pr(\, csets \mid \tau\,) \times \Pr(\, \tau \mid G, w\,)$$

To incorporate the semantic factor, both $\Pr(\, csets \mid \tau, G\,)$ and $\Pr(\, assignments \mid fillers, csets\,)$ need to be computed for each derivation τ.

As was shown in the previous chapter, computing $\Pr(\, assignments \mid fillers, csets\,)$ is straightforward given a semspec. It amounts to computing the fit of each filler to its semantic role. Figure 5.3 shows a derivation of "he slid" that admits two semspecs. In interpretation one, "he" is the mover, while in interpretation two, "he" is unbound and the mover role is unfilled. For interpretation one, the semantic factor would score the semantic fit of "he" filling the mover role, while for interpretation two, the semantic factor would score the semantic fit of "he" and the mover being unbound.[17] The distribution $\Pr(\, assignments \mid fillers, csets\,)$ models what I refer to as the "inner" probability of a semspec.

[17]The semantic factor as constructed in chapter 4 does not score unfilled roles, but it is an obvious extension that has been implemented in various iterations of the system.

Term $\Pr(\, csets \mid \tau, G\,)$ returns the likelihood of the semspec's structure given a derivation τ. In essence, it models the uncertainty over co-indexations in the feature structure given τ. For the example derivation in figure 5.3, $\Pr(\, csets \mid \tau, G\,)$ would define the likelihood of a semspec in which the profiledProtagonist role is bound to the mover role[18] and the likelihood of a semspec in which the profiledProtagonist is not bound to the mover role.[19]

For a complete constructional tree, all of the co-indexations are defined explicitly by the constructions used in the tree. Thus $\Pr(\, csets \mid t, G\,)$ for a complete τ is one because there is no uncertainty about how roles are co-indexed. But given a partial derivation τ of a complete tree, $\Pr(\, csets \mid \tau, G\,)$ plays a crucial role in computing the probability of an analysis. It functions as the link between the syntactic factor and the distribution over assignments $\Pr(\, assignments \mid fillers, csets\,)$. Distribution $\Pr(\, csets \mid \tau, G\,)$ returns what I refer to as the "external" probability of a semspec. Defining $\Pr(\, csets \mid \tau, G\,)$ requires building a distribution over semspecs that are admitted by derivation τ. The rest of this chapter is devoted to defining $\Pr(\, csets \mid \tau, G\,)$.

5.5.1 How Many Semspecs per Stack?

The derivation in figure 5.3 shows the interpretations associated with each step of a derivation. Notice though, that the simple stack

$$Root[S], Decl[FiniteVP], SlidePastTense[]$$

in derivation step 3 is compatible with a possibly infinite set of trees.[20] Yet it does not seem problematic to assume that there are only two interpretations of that stack. This is because most if not all of the constructions that can occur between the declarative (Decl) construction and the main verb "slid" have some function in relating the subject to a limited number of semantic roles.

[18] Which is approximately the likelihood of the SelfMotion construction

[19] Which is approximately the likelihood of the CauseMotion construction given the derivation

[20] For example, infinite application of the VP conjunction rule.

1. $Root[S], HePronoun[]$
   - Action: Propose $Decl \rightarrow Subj\ FiniteVP$
2. $Root[S], Decl[FiniteVP]$
   - profiledProtagonist (filled by "he")
   - Action: Push $SlidePastTense \rightarrow$ "*slid*"
3. $Root[S], Decl[FiniteVP], SlidePastTense[]$
   - INTERPRETATION 1: profiledProtagonist $\longleftrightarrow$ mover (filled by "he")
   - INTERPRETATION 2: profiledProtagonist (filled by "he")
   - Action: Propose $Trans \rightarrow Verb\ NP$
4. $Root[S], Decl[FiniteVP], Trans[NP]$
   - profiledProtagonist $\longleftrightarrow$ causer (filled by "he")
   - Action: Push $TheDet \rightarrow$ "*the*"
5. $Root[S], Decl[FiniteVP], Trans[NP], TheDet[]$
   - profiledProtagonist $\longleftrightarrow$ causer (filled by "he")
   - Action: Propose $DetNoun \rightarrow Det\ N$
6. $Root[S], Decl[FiniteVP], Trans[NP], DetNoun[N]$
   - profiledProtagonist $\longleftrightarrow$ causer (filled by "he")
   - Action: Push $BoxN \rightarrow$ "*box*"
7. $Root[S], Decl[FiniteVP], Trans[NP], DetNoun[N], BoxN[]$
   - profiledProtagonist $\longleftrightarrow$ causer (filled by "he"); mover (filled by "box")
   - Action: Attach $BoxN$ to $DetNoun.N$

Figure 5.3: The steps (up through the final push) that a left corner parser might go through for the sentence, *he slid the box*, now shown with relevant semantic bindings. The semantics do not change after the last operation shown. Notice that pushing "slid" leads to two possible interpretations.

Obviously, in the worst case, one could construct a grammar in which an infinite number of interpretations would need to be stored along with a stack. Such a scenario would undo the space savings gained by using stacks in the first place. But if one assumes that grammars have regularities in how they build interpretations, there will be a lot of shared semantic structure between the many parse trees that a stack supports. So far, the grammars actually in use have the kinds of semantic regularities that limit the number of semspecs associated with a stack.

5.5.2 The CR Relation

To use the semantic factor to score a derivation, the algorithm needs to infer a set of possible semspecs given the derivation. As we have seen, a left corner derivation can be represented by a stack ψ. Given the structure of a stack ψ, the algorithm infers the set of semspecs admitted by ψ by computing the likelihood of the co-indexation sets *csets*. Each *csets* assignment corresponds to a set of co-indexation assignments between a set of roles. Co-indexations between frame roles make up the structure of a semspec, and given the structure of a semspec it is straightforward to infer $\Pr(\, assignments \mid fillers, csets\,)$.

The process of building the possible semspecs admitted by a stack ψ is modeled as the pairwise information flow between the items on stack ψ. i.e. flow between α_{j-1} and α_j. The way the semantic roles are co-indexed between any arbitrary α_{j-1} and α_j can be precomputed using a recurrence relation CR that is similar to the computation of the probabilistic reachability relation R.

The *co-indexation record table* or CR is based on a recurrence relation that tracks all the different ways the roles between a construction α and construction θ can be co-indexed when θ is a descendant of α through constituent $\alpha.\beta$. A *co-indexation record* between α and θ is a list of slot chain pairs where one slot chain of each pair is rooted in α and the other is rooted in θ. (e.g. (($\alpha.role3$, $\theta.role5$) ($\alpha.role1$, $\theta.role4$))) Entry $CR(\alpha.\beta, \theta)$ in the co-indexation record table returns a list of co-indexation records along with an estimate of the probability of each co-indexation record.

The recurrence relation defining the CR table is:

$$CR(\alpha.\beta, \theta) = \delta(\alpha.\beta, \theta) \cup \forall_{\gamma \in G} \forall_{\lambda \in \gamma} (CR(\alpha.\beta, \gamma) \cap^* CR(\gamma.\lambda, \theta)) \quad (5.45)$$

$$\rho_0 \cap^* \rho_1 = \forall_{r_0 \in \rho_0} \forall_{r_1 \in \rho_1} J(r_0, r_1) \quad (5.46)$$

$$J(r_0, r_1) = \forall_{(\alpha.role, \gamma.role_0) \in r_0} \forall_{(\gamma.role_1, \theta.role) \in r_1} \; if\ role_0 = role_1\ then\ (\alpha.role, \theta.role) \quad (5.47)$$

The base case of the CR relation defined in equation 5.45 is the δ function which unifies θ into role $\alpha.\beta$ and returns a co-indexation record based upon direct attachment. This co-indexation record consists of slot chains rooted in α and slot chains rooted in θ that are unified. The probability of the co-indexation record is the likelihood that θ is an expressed, local filler of β or:

$$\Pr(\, expressed_\beta \mid \alpha \,) \times \Pr(\, local_\beta \mid expressed_\beta, \alpha \,) \Pr(\, filler = \theta \mid expressed_\beta, \alpha \,)$$

The recursive part of the CR relation in equation 5.45 looks for indirect connections between $\alpha.\beta$ and θ by enumerating all possible intermediaries (γ) and each constituent λ of γ. If there is a co-indexation record between $\alpha.\beta$ and γ and another co-indexation record between $\gamma.\lambda$ and θ, then there is a co-indexation record between $\alpha.\beta$ and θ through γ.

The $\cap^*$ operator in equation 5.45 performs this modified intersection between the co-indexation records of $CR(\alpha.\beta, \gamma)$ and $CR(\gamma.\lambda, \theta)$. Assume $CR(\alpha.\beta, \gamma) = \rho_0$ and $CR(\gamma.\lambda, \theta) = \rho_1$. Then for each co-indexation record pairing (r_0, r_1) where $r_0 \in \rho_0$ and $r_1 \in \rho_1$, a join operation (the J operation in equation 5.47) computes the new co-indexation record by joining the second element of the r_0 pairs with the first element of the r_1 pairs.

The probability of the new co-indexation record is the product of r_0's probability with r_1's probability times the probability of using constituent λ of construction γ. This can computed as the product of omitting all the constituents in $precedes(\lambda)$. If the new co-indexation record already exists, then this probability is added to the pre-existing probability of the co-indexation record.

Additionally, no connection between the two types is also a co-indexation record that indicates the chance that no roles are shared between α and θ when going through β. No connection is the most common kind co-indexation record, especially when the minimum path between the constructions gets long.

5.5.3 An Example

An example of CR table is shown in figure 5.5. The table in figure 5.5 is based on the grammar in figure 5.4. The grammar in figure 5.4 has three kinds of constructions X, Y and Z. The Z constructions are the "lexical items". The X construction is the root, and the Y constructions are the "phrases" of the grammar. The Y constructions act as semantic intermediaries between the semantic roles of the lexical items and the semantic roles of the root.

Depending on the chain of constructions used, different roles are co-indexed. The table in figure 5.5 shows how the roles of the different constructions through direct constituency (with path length zero), and additionally through a Y construction (path length one). Building the two path length 1 entries of the table requires the recursive step from the co-indexation record algorithm. If the table specifies the co-indexation record as "no connection", that means that no roles of the two constructions are co-indexed in that particular case. Connection records that connect the exact same sets of roles are merged together. For example, the entry (X, Z1, 1) in the co-indexation record table in figure 5.5 has only three records instead of four. This is because the "no connection" record covers two different paths. Whenever a single connection record covers multiple paths, the probabilities for each path are added together to determine the likelihood of the entry.

5.5.4 Building Semspecs from a Stack Using the CR Records

The intuition behind building the complete semspec for a stack is to start by associating each item on the stack with a feature structure that represents that part of the semspec that the item somehow constrains. Then using the co-indexation records, a single consistent semspec is built up.

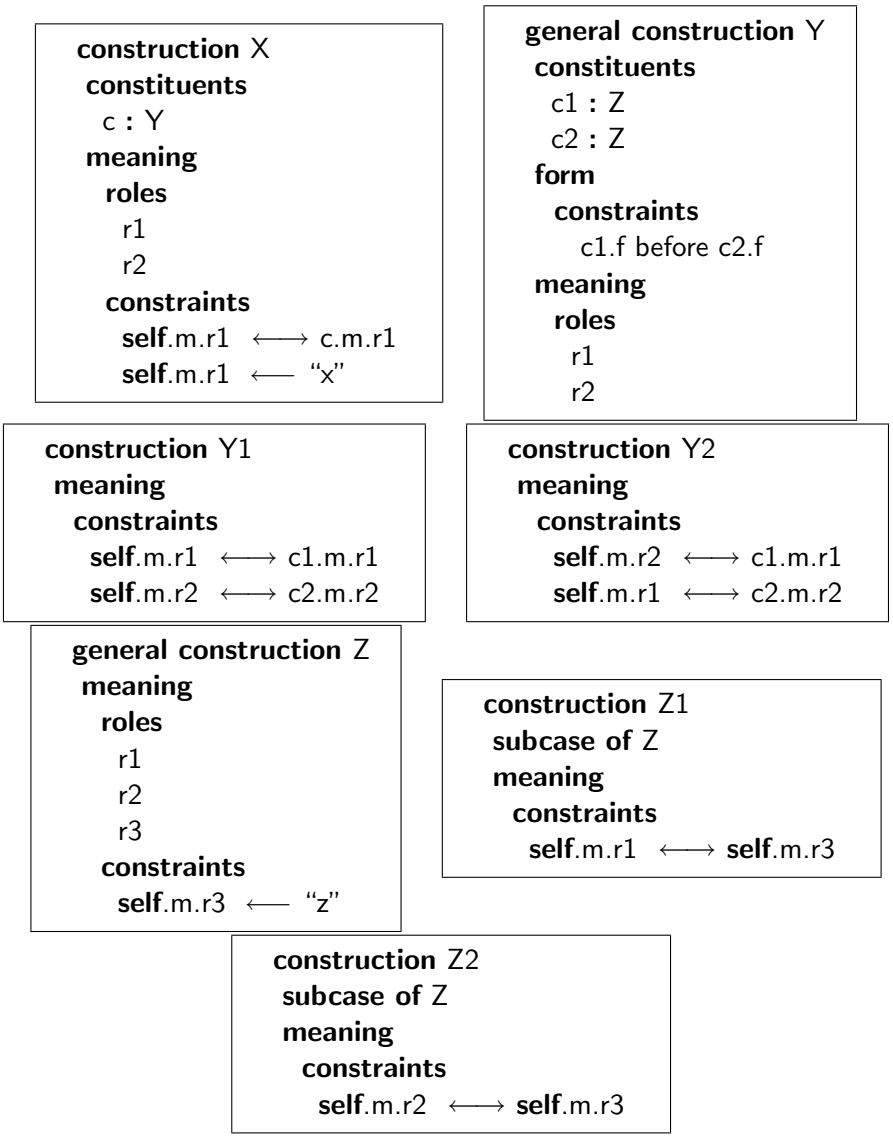

Figure 5.4: A simple example grammar for use in illustrating the process of building co-indexation records. This grammar admits eight parse trees: (X (Y1 Z1 Z1)), (X (Y1 Z1 Z2)), (X (Y1 Z2 Z1)), (X (Y1 Z2 Z2)), (X (Y2 Z1 Z1)), (X (Y2 Z1 Z2)), (X (Y2 Z2 Z1)) and (X (Y2 Z2 Z2)).

Cons	DT	PL	Record	Probability
X.c	Y1	0	X.m.r1 ⇔ Y1.m.r1	p_{xc}
X.c	Y2	0	X.m.r1 ⇔ Y2.m.r1	q_{xc}
Y1.c1	Z1	0	Y1.m.r1 ⇔ Z1.m.r3	p_{y1c1}
Y1.c2	Z1	0	Y1.m.r2 ⇔ Z1.m.r2	p_{y1c2}
Y1.c1	Z2	0	Y1.m.r1 ⇔ Z2.m.r1	q_{y1c1}
Y1.c2	Z2	0	Y1.m.r2 ⇔ Z2.m.r3	q_{y1c2}
Y2.c1	Z1	0	Y2.m.r2 ⇔ Z1.m.r3	p_{y2c1}
Y2.c2	Z1	0	Y2.m.r1 ⇔ Z1.m.r2	p_{y2c2}
Y2.c1	Z2	0	Y2.m.r2 ⇔ Z2.m.r1	q_{y2c1}
Y2.c2	Z2	0	Y2.m.r1 ⇔ Z2.m.r3	q_{y2c2}
X.c	Z1	1	X.m.r1 ⇔ Z1.m.r3 no connection X.m.r1 ⇔ Z1.m.r2	$p_{xc}l_{y1c1}p_{y1c1}$ $p_{xc}o_{y1c1}l_{y1c2}p_{y1c2} + q_{xc}l_{y2c1}p_{y2c1}$ $q_{xc}o_{y2c1}l_{y2c2}p_{y2c2}$
X.c	Z2	1	X.m.r1 ⇔ Z2.m.r1 no connection X.m.r1 ⇔ Z2.m.r3	$p_{xc}l_{y1c1}q_{y1c1}$ $p_{xc}o_{y1c1}l_{y1c2}q_{y1c2} + q_{xc}l_{y2c1}q_{y2c1}$ $q_{xc}o_{y2c1}l_{y2c2}q_{y2c2}$

- p_{xc} and q_{xc} are the probabilities that constituent X.c is filled by Y1 and Y2, respectively.
- p_{y1c1} and q_{y1c1} are the probabilities that constituent Y1.c1 is filled by Z1 and Z2, respectively. l_{y1c1} and o_{y1c1} are the local and omission probabilities of Y1.c1.
- p_{y1c2} and q_{y1c2} are the probabilities that constituent Y1.c2 is filled by Z1 and Z2, respectively. l_{y1c2} and o_{y1c2} are the local and omission probabilities of Y1.c2.
- p_{y2c1} and q_{y2c1} are the probabilities that constituent Y2.c1 is filled by Z1 and Z2, respectively. l_{y2c1} and o_{y2c1} are the local and omission probabilities of Y2.c1.
- p_{y2c2} and q_{y2c2} are the probabilities that constituent Y2.c2 is filled by Z1 and Z2, respectively. l_{y2c2} and o_{y2c2} are the local and omission probabilities of Y2.c2.

Figure 5.5: The CR table for the grammar in figure 5.4. The Cons column indicates the source constituent, the DT column indicates the descendant type , the PL column indicates the distance between the constituent and the descendant in terms of intermediate constructions, the Record column indicates how the semantic roles unify, the Probability column indicates the probability of the record. Horizontal lines in the table separate entries in the CR table. For example the first entry has exactly one connection record, while the entry for X.c, Z1, 1 has three. "no connection" indicates that no roles are connected.

Starting from the root of the stack, information is pushed from $item_{j-1}$ to $item_j$. For each co-indexation record ρ that connects $item_{j-1}$ with $item_j$ through constituent β, each partial semspec Φ associated with $item_{j-1}$ is unified with the partial semspec at $item_j$ according to the constraints in ρ. The (unnormalized) probability of the new partial semspec is the probability of the original semspec Φ times the probability of the co-indexation record ρ times the probability of omitting[21] the unmatched predecessors of β:

$$\Pr(\,\Phi \mid item_{j-1}\,) \times \Pr(\,\rho \mid item_{j-1}, item_j, \beta\,) \times P_{omit(\beta)}$$

The maximum number of resulting partial semspecs at $item_j$ after this process is the number of partial semspecs at $item_{j-1}$ times the number of co-indexation records connecting the two items. It is only the maximum number because some of the unifications for a given ρ and Φ could fail, which would require re-normalization.

As an example, consider the following stack based on the grammar in figure 5.4: $X[c : Y], Z1[]$. Before any information is passed between stack items, the partial semspec for X looks like $[X.m.r1 : x, X.m.r2 :]$ and the partial semspec for $Z1$ looks like $[Z1.m.r2 :, Z1.m.r3 : z]$. According to the co-indexation record table in figure 5.5, there are three co-indexation records between X and $Z1$. The first connects $X.m.r1$ with $Z1.m.r3$ and leads to a unification failure. The second record has no connections, so the resulting semspec looks like $[X.m.r1 : x, X.m.r2 :, Z1.m.r2 :, Z1.m.r3 : z]$. The third co-indexation record connects $X.m.r1$ with $Z1.m.r2$ which results in a semspec that looks like $[X.m.r2 :, Z1.m.r2 : x, Z1.m.r3 : z]$.

The feature structures associated with the top of the stack are the feature structures consistent with the complete derivation. Thus the normalized[22] probability of one of the complete feature structures is $\Pr(\,csets \mid t, G\,)$ for one assignment to *csets* (the external probability of a semspec). It is this probability that relates the syntactic factor and the semantic factor.

[21] or setting exactly one to be nonlocal

[22] Normalizing deals with unification failures and defines a proper distribution over csets

5.5.5 Updating the Left Corner Parser Operations

Of the three main left corner parser operations, only the attach operation requires any thought. A push operation just puts a new item on the top of the stack, and therefore the algorithm for pushing semantic information up the stack described in section 5.5.4 already does the necessary work. And from the point of view of these feature structure updates, a propose action is just a combination of a push and an attach.

When an attach operation takes place, semantic information must be pushed from a child α_{j+1} to its new parent α_j. The attach takes place between α_{j+1} and a particular constituent β_j of α_j. α_j has k_j feature structures (interpretations) associated with it, and there are at most k_j feature structures in $\alpha'_{j+1}s$ feature structure set that have the relevant information to be pushed back to α_j. At most k_j, because the co-indexation CR table entry has exactly one direct attach co-indexation record[23] for β_j and α_{j+1}, and that co-indexation record would lead to k_j feature structures in α_{j+1}.

Given the appropriate co-indexation record for a direct attach, each of the k_j parent feature structures receives updated information from its corresponding child structure should that corresponding child structure still exist. If the corresponding child structure does not exist, then that means that the corresponding child feature structure suffered a unification failure during the lifetime of item α_{j+1}, and the corresponding parent feature structure gets removed from α_j. The external probabilities of each of the remaining feature structures in α_j must then be re-normalized to account for the possibility of unification failure.

5.6 Summary

This chapter defined a probabilistic model of interpretation based on left corner parsing. It is compatible with the factored syntax and semantic model defined in the previous chapter. The method for building up coherent semantic interpretations given

[23]The direct attach co-indexation record is the one indexed by path length 0 in the co-indexation record table.

a stack allows for semantic information to condition parser choices without relying on a fully connected parse tree. The value of the probabilistic model of interpretation defined here is shown in the next three chapters. Those case studies show both the value of the probabilistic model for improving parser efficiency, but also to make detailed reading time predictions for experimental reading time data.

Chapter 6

Deep Linguistic Representation with the Constructional Analyzer

This chapter shows how a combination of ECG and the constructional analyzer can be used to help design theories of embodied semantics and how solutions to classical problems in syntax can be implemented. In essence, this case study is the qualitative baseline for the system. All the example semspecs shown in this chapter are produced by the analyzer using a single grammar. For lack of a better name, we call this grammar EJ1 which is short for Ellen/John Grammar 1. To see the rest of the schemas and constructions of EJ1 that are not shown here, visit the ECGwiki.[1]

While construction grammar has a rich tradition in linguistics, computational research on construction grammar has moved at a slower pace. Apart from my own previous work [5], the work of Jurafsky[2] [34], and research with Fluid Construction Grammar [79], there has not been computational research that focuses on construction grammar. As a consequence, the grammar subset described in section 6.1 is the first "true" construction grammar used by a parser to interpret a sentence. Of course, there have been earlier grammars written in various construction grammar formalisms, but these grammars did not leverage deep semantics and were not compositional with constructions that represented interesting linguistic phenomena such as passive,

[1] http://ecgweb.pbwiki.com/

[2] Broadly speaking, my work can be viewed as the next generation of Jurafsky's work, except that I reap the benefits of 15 more years of construction grammar research.

raising, control, and radial categories. Section 6.1 reports on the Dodge's insights on how cause-effect and impact semantics are combined using verbs and argument structure constructions.

Section 6.2 shows how to make the argument structure constructions in section 6.1 more productive. It explains how they might fit together with constructions that alter how semantic arguments are expressed. The work in section 6.2 is joint work with Ellen Dodge.

In section 6.3, I show how to take the basic ideas described in sections 6.1 and 6.2 and apply them to the ditransitive (double object) construction. The radial category structure of the ditransitive was described by Goldberg [26], however, her work suffers from incomplete productivity and lacks a proper formalism. My treatment shows how ECG can be used to further develop her approach.

Apart from the work of Dodge described in section 6.1, the analyses suggested in this chapter are proof of concept analyses. They show that the constructional analyzer supports subtle grammatical and semantic inference, and can produce cognitively-motivated analyses of the same level of complexity that the HPSG parser [87] or the LFG parser [66] can generate.

6.1 Embodied Schemas

Dodge [16] models actions and events using three basic schemas: Process, Complex-Process and MotorControl shown in figure 6.1. Process and ComplexProcess are general descriptions of actions and events in which a single participant is profiled using the protagonist role. A process that is treated as atomic is a subcase of Process, but not a subcase of ComplexProcess. The Motion schema shown in figure 6.2 is treated as atomic. The Motion schema describes a process in which the mover is the protagonist and has a speed and heading. The MotionAlongAPath schema is a subcase of Motion and adds the constraint that the motion is conceptualized as occurring along a path. The path is represented by the evoked SPG schema shown in chapter 2. In both schemas, the mover is defined as the protagonist using a co-indexation constraint.

Each process has a role called x-net that further specifies the kind of action that

is modeled by the schema. For example, the Motion process shown in figure 6.2 can describe both walking and crawling (and many other methods of motion). Walking and crawling have very different motor actions involved, but the assumption is that from the point of view of the grammar, those specific aspects of walking and crawling can be represented by the filler of the xnet role. As a consequence, the Motion schema acts as an abstraction over all the different motion x-nets. The grammar writer can use the generic parameters encoded by the Motion schema without necessarily having to encode x-net specific information in the grammar.

Schema MotorControl has a special significance in the grammar. It is the semantic root of embodied, controlled processes. It adds roles for actor, effector, effort and routine. The actor is the embodied protagonist, the effector is the controlled body part, the effort is the energy expenditure and the routine role is used to refer to embodied motor control x-nets.

A decomposable process is represented by ComplexProcess is a process that characterizes the interaction between a pair of processes labeled with roles process1 and process2. For example, causing motion is a kind of complex process because it combines a force application with a motion in which the recipient of the force is also the mover in the motion. Transfer is also a complex process because one participant is relinquishing an object while the other is taking it. The two subprocesses of a complex process can share any kind of temporal relation as long as they are conceptualized jointly as part of larger gestalt.

Most verbs in the EJ1 grammar specify a process schema as their meaning and then bind lexically specific parameters with their meaning blocks. For example the SlidePastTense construction (also shown in figure 6.2) specifies that its meaning is a MotionAlongAPath schema. It also specifies that the x-net role is filled by a slide.[3]

In ECG, and any construction-based theory of grammar for that matter, verbs are combined with their arguments using argument structure constructions. ECG also relies on a semantic intuition taken from the cognitive linguistics community [26] that an argument structure construction specifies the scene being described while the verb

[3]Remember that the @ symbol indicates that the type is defined in an external ontology.

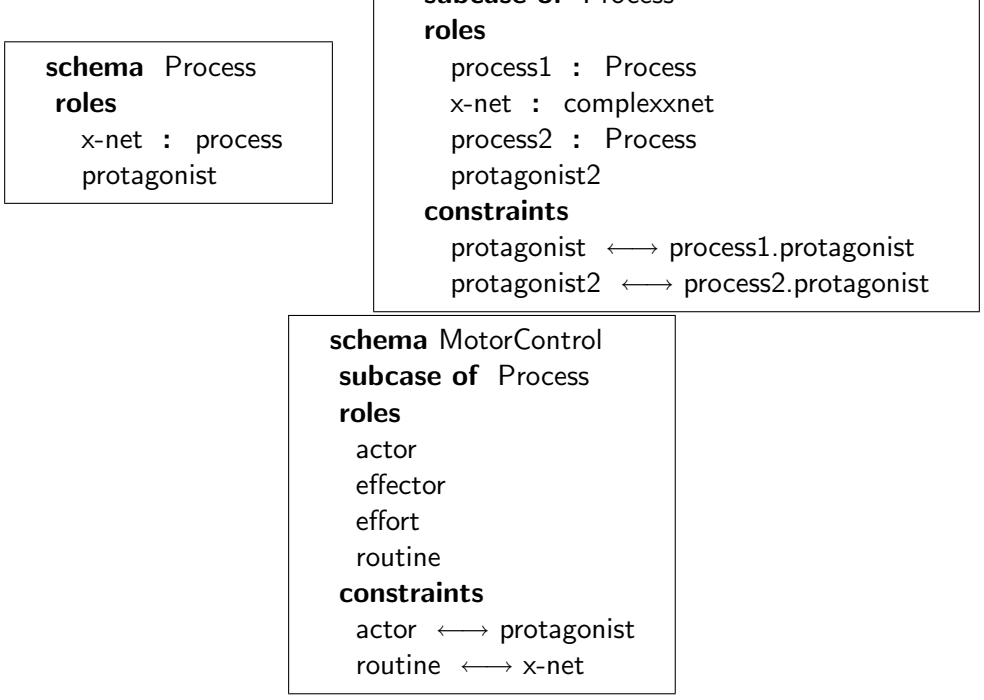

Figure 6.1: The Process and ComplexProcess schemas

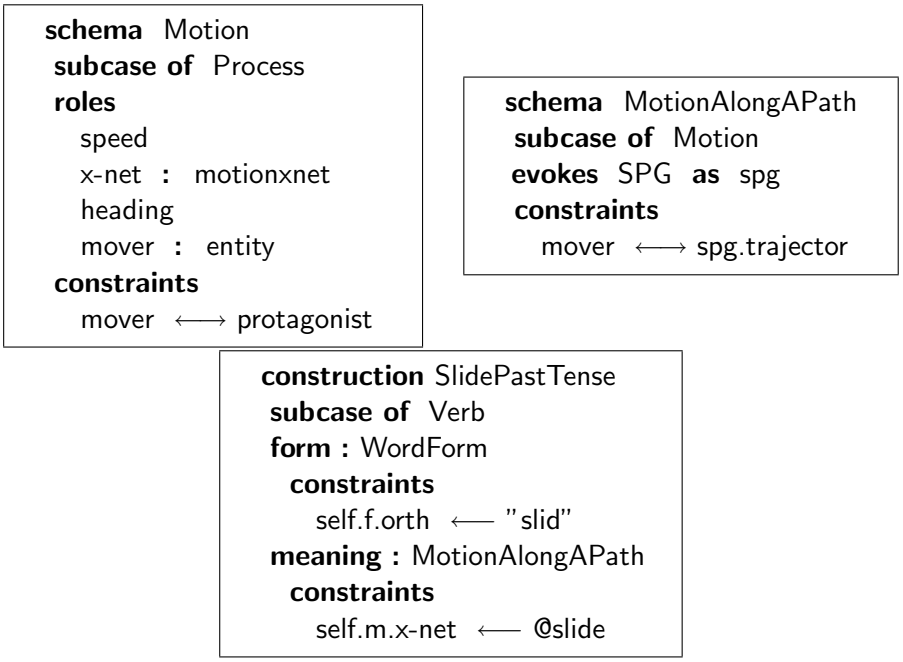

Figure 6.2: Motion and MotionAlongAPath schemas along with a lexical construction SlidePastTense for the word "slid".

elaborates the means by which the action is carried out. In central-case argument structure constructions, the verb and the argument structure construction's meanings line up perfectly (e.g. He gave me a book), while non-central argument structure constructions combine the semantics of the verb with the semantics of the scene in a motivated, but not necessarily predictable way.

A further claim that ECG makes is that argument structure constructions also provide guidance to the simulation about how a scene should be simulated. For example, active and passive provide differing perspectives on the same scene, and such a perspective shift must be communicated to the simulator. The EJ1 grammar uses a schematic abstraction over scenes (events) called the EventDescriptor schema (shown in figure 6.3) to represent perspectivized facts about a scene. The EventDescriptor schema has more roles than shown in figure 6.3, but the roles that are shown are useful for this chapter. The roles that I have included are:

- The eventType role is bound to the Process that represents the scene being described. The argument structure construction supplies the filler of this role.
- The profiledProcess role is bound to the subprocess in the scene that is being profiled. The verb supplies the filler of this role.
- The profiledParticipant role is bound to the participant (entity) in the scene that is being profiled. This role can be thought of as the semantic correlate of subject, and is bound to different roles in a scene depending on whether the utterance is active or passive voice.
- Roles temporalSetting and locativeSetting are bound to the time and location of the scene, respectively.
- The discourseSegment role is typed to a DiscourseSegment schema. The extremely simplified DiscourseSegment schema has roles for the speechAct of the utterance and the topic of the utterance. In this chapter, the speechAct role will be bound to an atomic value such as "declarative" or "wh-question". The topic role will be bound to the topic specified by each finite clause.

Figure 6.3 shows how a general VerbPlusArguments construction and the MotionAlongAPath construction co-index their roles with the EventDescriptor. The VerbPlusArguments construction is a special kind of VP construction that has a Verb constituent.[4] Its verb constituent's meaning is bound to the profiledProcess role of its evoked EventDescriptor. The MotionAlongAPath construction inherits the evoked EventDescriptor and a constraint between the verb and profiledProcess role. It also adds a constraint that the mover role in the MotionAlongAPath scene is bound to the profiledParticipant. As will be shown in the next section, the S-With-Subj construction binds the subject to the profiledParticipant role and thereby fills the appropriate semantic role in the scene.

Figure 6.5 shows the schemas and constructions for cause-effect transitive sentences like, *He hit the table* and *The hammer hit the table.* According to Dodge, both of those sentences evoke a ForcefulMotionAction schema through the word *hit.* A ForcefulMotionAction is a ComplexProcess in which the protagonist applies a force (process1) to a target via the motion of an effector (process2). The ForceApplication schema (not shown) is a MotorControl schema in which the effector applies a force to the actedUpon. The EffectorMotionPath schema (also not shown) is a kind of ComplexProcess that combines a MotorControl with MotionAlongAPath, binding the MotorControl.effector to the MotionAlongAPath.mover.[5] The constraints in ForcefulMotionAction state that ForceApplication.actedUpon is the same as the EffectorMotionPath.target and the instrument of the force is the same as the effector of the EffectorMotionPath.

Dodge also suggests that the meaning of the central transitive construction involves a notion of causation. As is shown in figure 6.5, she encodes the central notion of causation with the CauseEffectAction schema. A CauseEffectAction is also a complex process where process1 is a ForceApplication acting as the cause and process2 refers to the effect. The CauseEffectAction schema co-indexes its causer role with the protagonist role (which is also process1.protagonist), the affected role to the protagonist2 role (which is also process2.protagonist). It also binds process1.actedUpon with its affected role to make the ForceApplication apply to the filler of the affected role.

[4] A VP construction could also be a conjunction of two VPs.

[5] For more information about ForceApplication and EffectorMotionPath, see Dodge's dissertation [16].

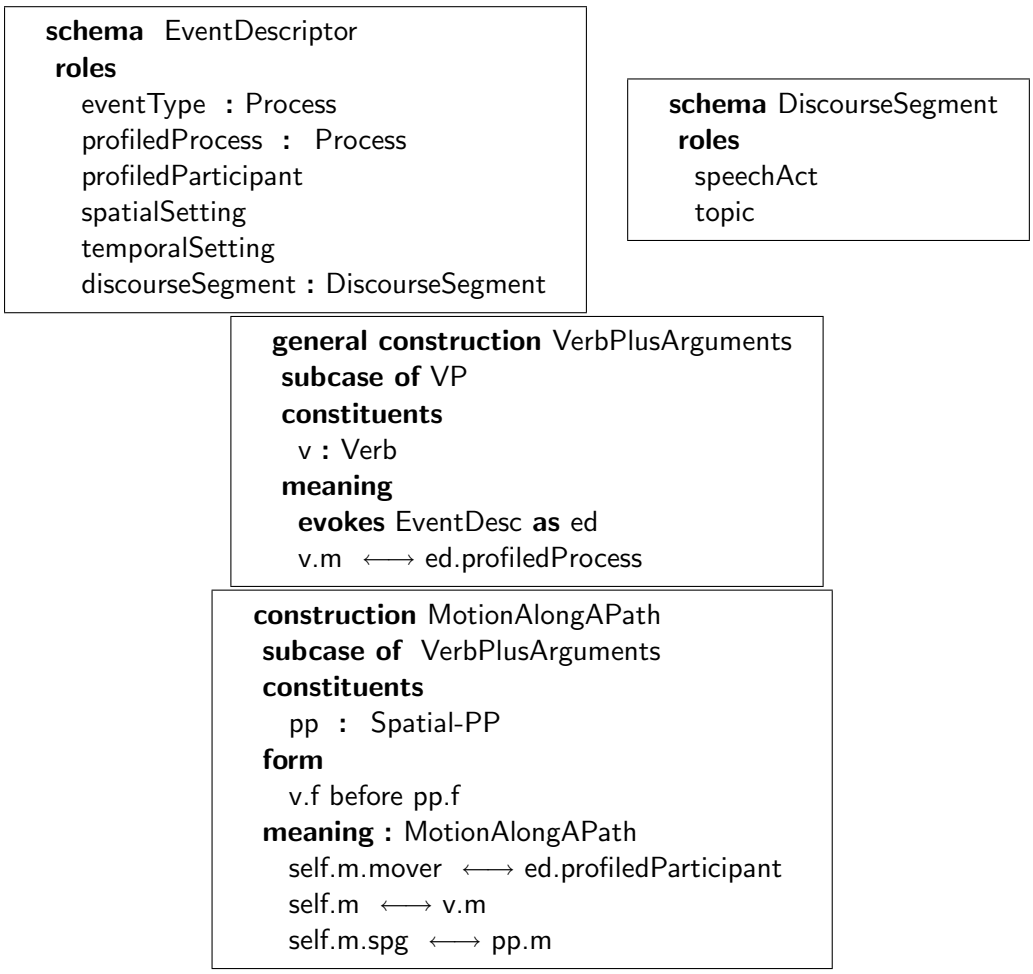

Figure 6.3: An example EventDescriptor schema, a general VerbPlusArguments construction and the VP MotionAlongAPath representing a basic motion verb phrase such as "walked into the house"

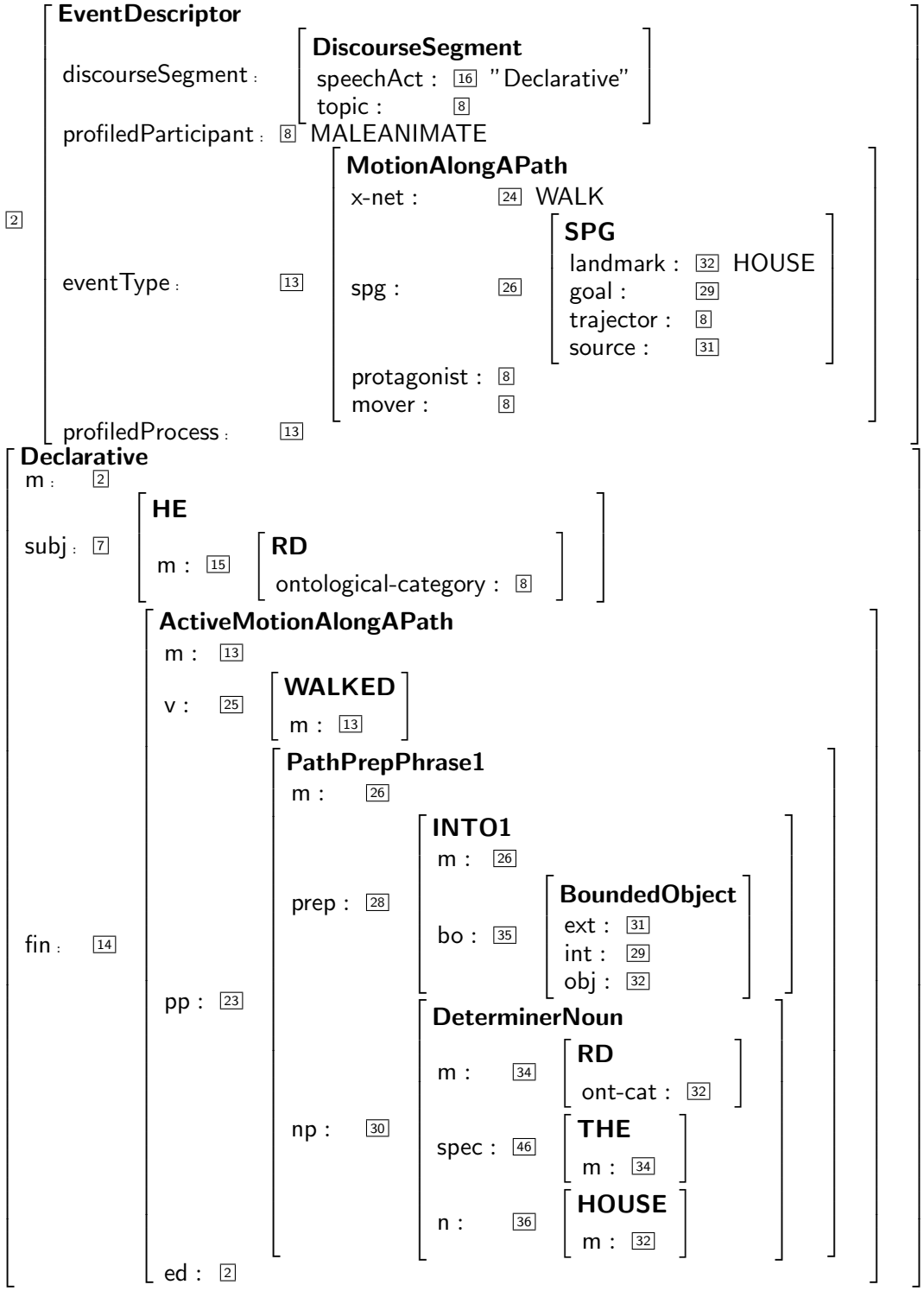

Figure 6.4: The analysis for *he walked into the house*.

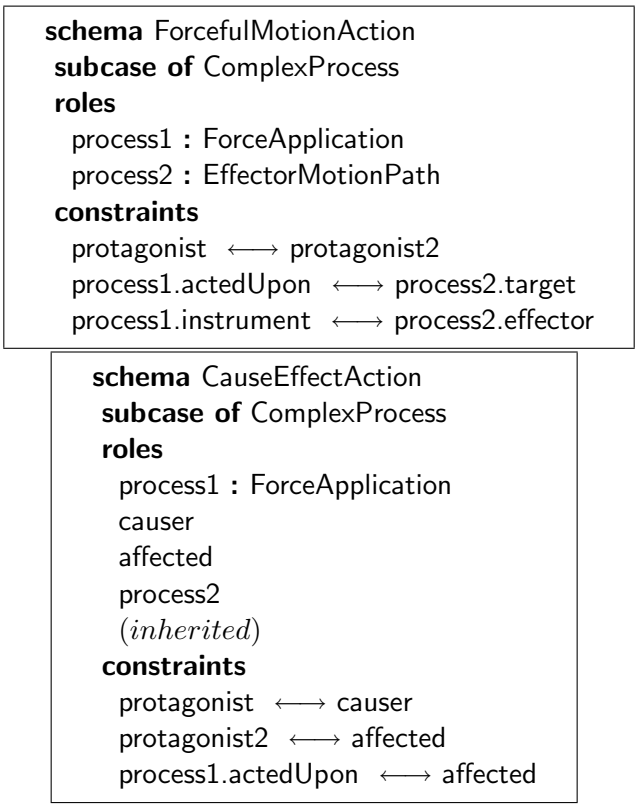

Figure 6.5: The ForcefulMotionSchema represents the application of force at some target via the motion of an effector. The CauseEffectAction is a ComplexProcess that models a cause-effect relation between two processes where process1 is a kind of ForceApplication.

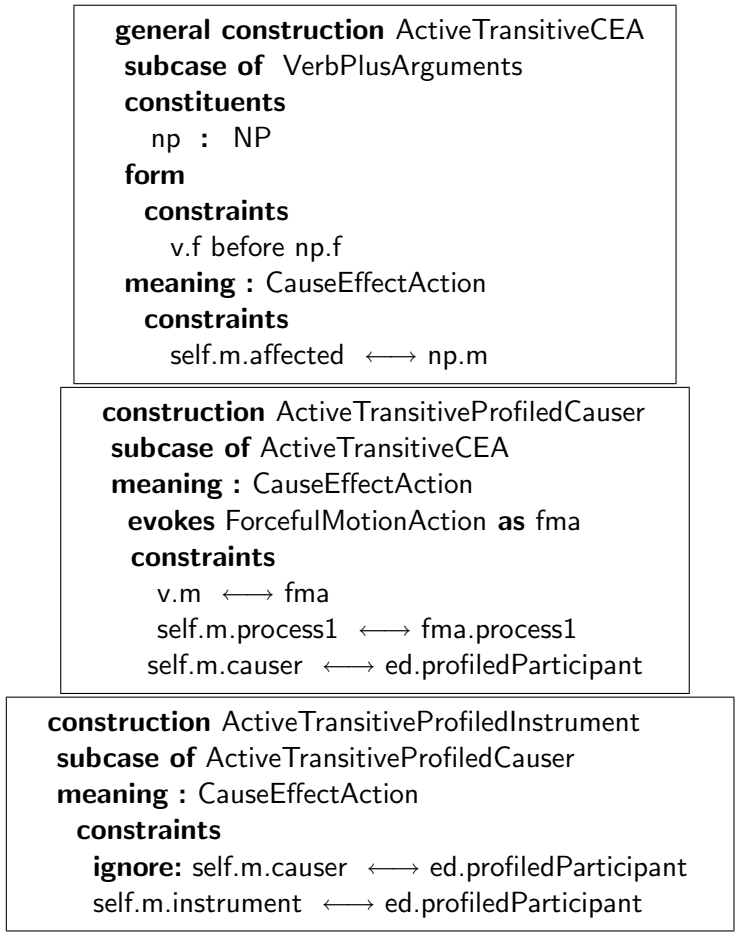

Figure 6.6: The ActiveTransitiveCEA construction is a general transitive construction that uses the CauseEffectAction as its meaning. The concrete constructions ActiveTransitiveProfiledCauser and ActiveTransitiveProfiledInstrument each impose a perspective on the basic transitive scene, either profiling the causer or the instrument, respectively.

Dodge shows how the CauseEffectAction schema can be used to define the meaning of a central transitive construction. This general central transitive is ActiveTransitiveCEA, shown in figure 6.6. The ActiveTransitiveCEA is a subcase of VerbPlusArguments that combines the inherited verb constituent with an NP constituent acting as the object. Because the construction is in active voice, the form block constrains the verb to be before the object NP. Dodge sets the meaning of the ActiveTransitiveCEA to the CauseEffectAction, adding a constraint that binds the affected role to the meaning of the object NP.

The ActiveTransitiveCEA is a general construction because it does not specify how the meaning of the verb combines with the CauseEffectAction. Two concrete subcases of ActiveTransitiveCEA are also shown in figure 6.6. The ActiveTransitiveProfiledCauser handles the VP in *he hit the table*, and so it evokes a ForcefulMotionAction, co-indexing it with the meaning of the verb. This limits the application of the ActiveTransitiveProfiledCauser construction to verbs like, "hit". The second constraint unifies the process1 of the construction's CauseEffectAction with the evoked ForcefulMotionAction.process1, thereby unifying the ForceApplications of the two schemas. The third constraint co-indexes the causer role with the evoked EventDescriptor's profiledParticipant role thereby asserting that the ActiveTransitiveProfiledCauser does indeed profile the causer.

The ActiveTransitiveProfiledInstrument is a special transitive that profiles the instrument. It is applied to sentences like, "the hammer hit the table". ActiveTransitiveProfiledInstrument is a subcase of ActiveTransitiveProfiledCauser that ignores the binding between the causer and the profiledParticipant roles. The construction instead unifies the instrument role with evoked EventDescriptor's profiledParticipant role.

The important point to take away from the constructions in figure 6.6 is how much shared structure there is between ActiveTransitiveProfiledCauser and ActiveTransitiveProfiledInstrument. Building the argument structure constructions in this way motivates the similarity between *he hit the table* and *the hammer hit the table*. Everything about the two usages is the same except for a single binding. Thus her solution is simpler, better motivated and more elegant than a Goldberg style approach to argument structure constructions [26]. Furthermore, Dodge's solution uses deep semantics to indicate which verbs can be a constituent of these two constructions. Using se-

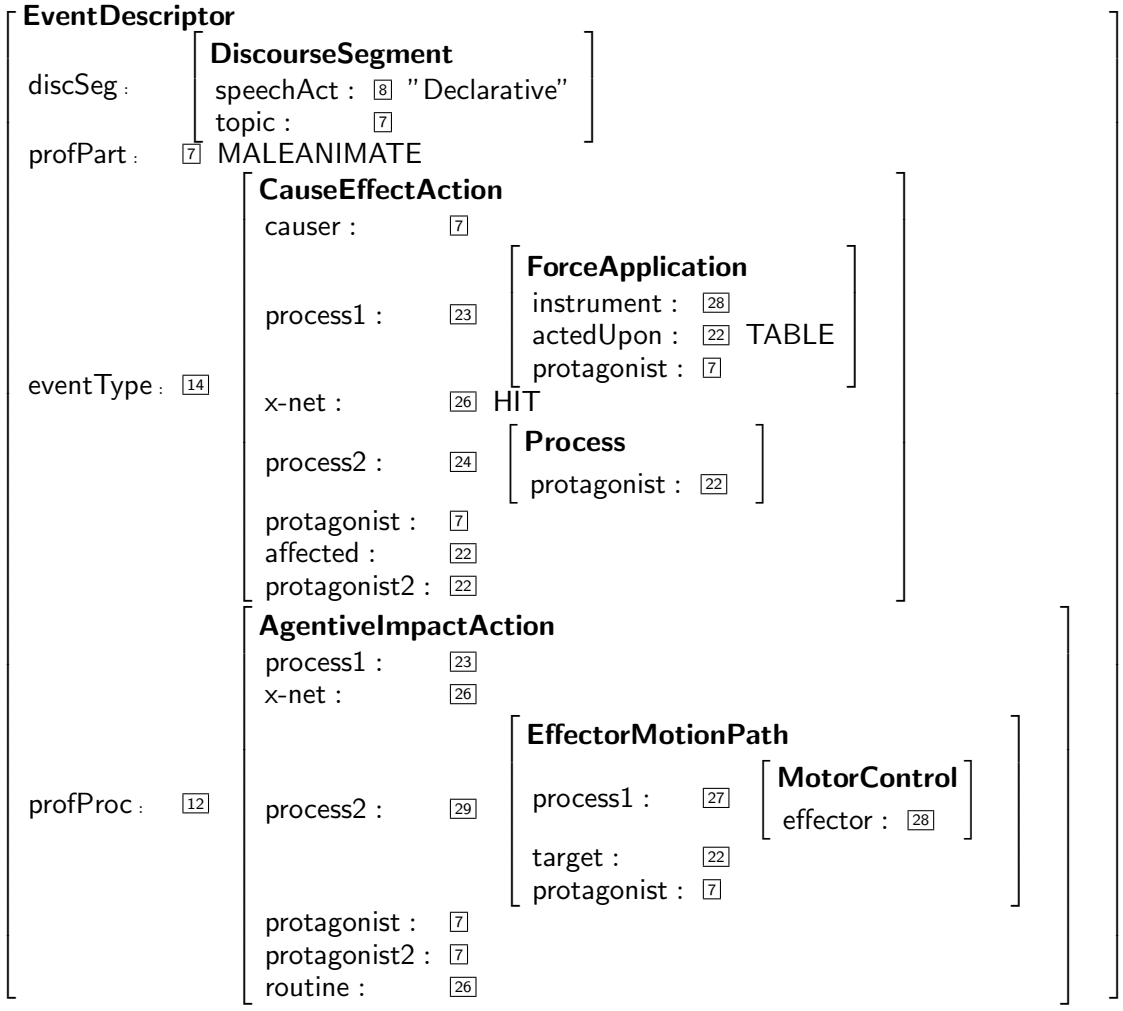

Figure 6.7: The semspec for the sentence *he hit the table*.

mantics is elegant because it builds on a semantic substrate that is already necessary for inference. Thus her theory does not rely directly on the lexicon.

6.2 Improving Compositionality

The previous section showed how embodied semantics could be integrated into a construction grammar using schemas and argument structure constructions. To apply the constructions from the previous section (and the rest of the constructions

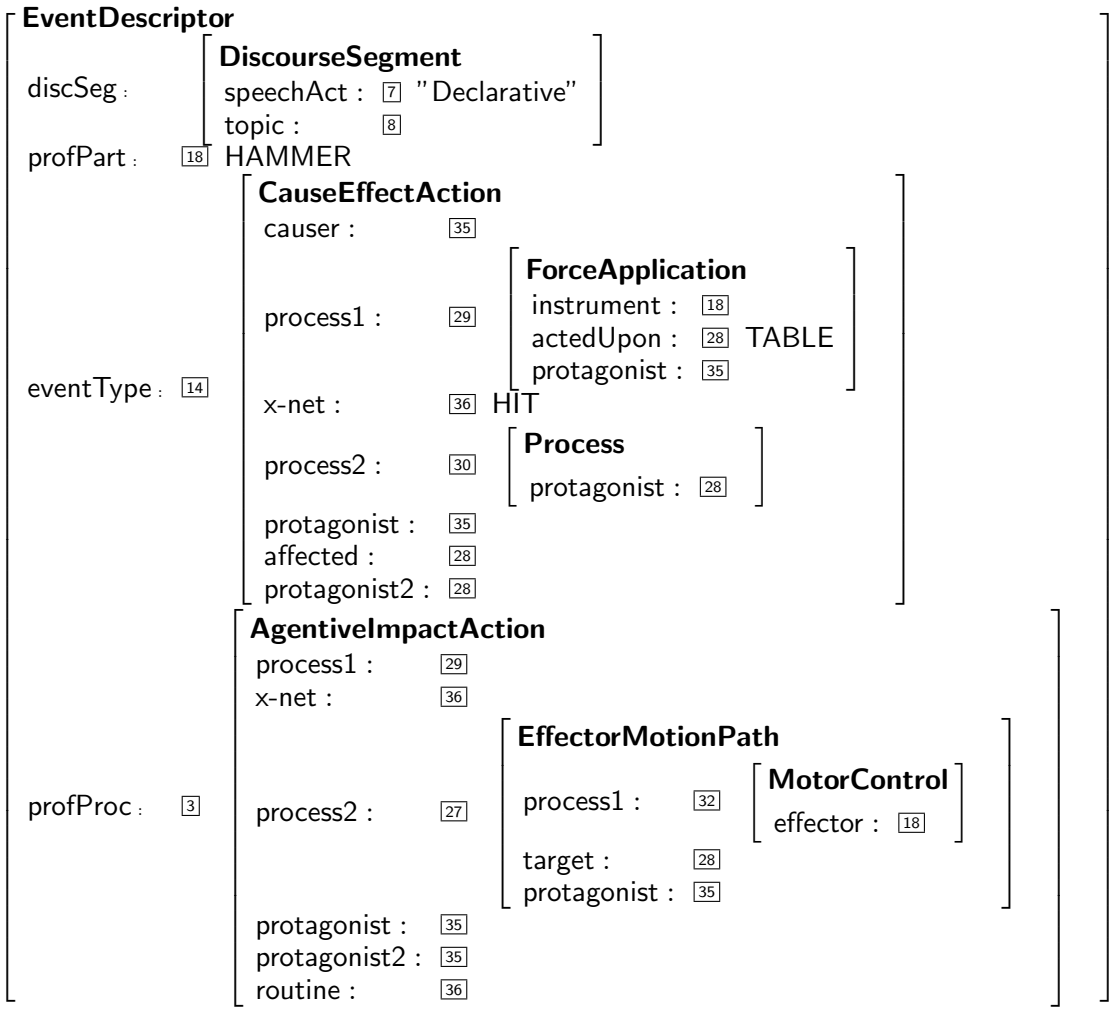

Figure 6.8: The semspec for the sentence *the hammer hit the table.*

described by Dodge) to a broader range of linguistic data, the argument structure constructions must compose with constructions for different kinds of speech acts. This section shows proof of concept constructions to model passive, subject-verb agreement, questions, and object raising.

6.2.1 Passive

The question of how active and passive are related is longstanding question in linguistics. I have nothing insightful to add to this discussion beyond implementing a solution that treats active and passive as different families of constructions related through common semantics. The motivation for such a choice is beyond the scope of this dissertation, except for the following additional computational motivation for such a choice: Active and passive constructions are not used with the same frequency, and having unique constructions makes it obvious how one could represent these statistical differences.

The basic idea behind these constructions is to use the exact same semantic schemas as the active counterparts, while inheriting the form constraints through the passive hierarchy. In this case study, the top of the passive hierarchy is the general construction PassiveCEA. Then the only thing that concrete subcases of PassiveCEA need to do is specify what kinds of verbs are allowed (via meaning constraints) and how the semantics of the optional by-phrase are bound into the semspec.

Figure 6.9 shows the passive versions of the ActiveTransitiveCEA constructions shown in figure 6.6. Construction PassiveCEA is subcase of VerbPlusArguments, which means it inherits a verb constituent. PassiveCEA adds an optional ByPhrase and requires that it come after the verb. Semantically, PassiveCEA denotes a CauseEffectAction, and unifies the affected role with the evoked EventDescriptor.profiledParticipant. This difference in binding is exactly the difference between active and passive in the EJ1 grammar. The profiledParticipant in the active case is the causer, while in the passive case, the affected participant is the profiledParticipant.

The concrete subcases of PassiveCEA are similar to their active counterparts. PassiveCEACauserByPhr evokes a ForcefulMotionAction, binds the two ForceApplications to-

gether, and then unifies the meaning of the ByPhrase with the causer. The Passive-CEAInstrumentBYPhr inherits all the structure of PassiveCEACauserBYPhr, but instead of binding the causer to the meaning of the ByPhrase, it unifies with the instrument role.

Figure 6.10 shows the semspec that is generated by the analyzer for the passive sentence, *The table was hit by the hammer.* Comparing this semspec to the semspec for *the hammer hit the table* shown in figure 6.8. The two sentences describe the same scene and instantiate the same frames. However the passive utterance profiles the table, while the active sentence profiles the hammer. This difference is cashed out in different bindings for the profiledParticipant role. The passive sentence co-indexes the table to that role, while the active sentence binds the hammer to the role.

6.2.2 Agreement

Subject and main verb agreement is the paradigmatic example for illustrating the power of unification grammar. Because ECG is a kind of unification grammar, it should come as no surprise that agreement is straightforward to implement in the EJ1 grammar. The general construction S-With-Subj shown in figure 6.11 defines a grammatical category that has an NP subj constituent and a finite constituent of type HasVerbFeatures. The general construction HasVerbFeatures is the supertype of verbs as well as VPs, and it types the constructional block to have features like number, person and verbform. The general NP construction also specifies its constructional type to be HasNominalFeatures which adds features for person, number and case.

The constraints block of construction S-With-Subj shows how these features are used. The first constraint ensures that constituent fin is a finite form. In this formulation, both VPs and verbs have verbal features. If a VP has a main verb that is a finite form, then the VP will also be marked as finite. Thus fin refers to features of the main verb of the S-With-Subj construction. The next two constraints enforce agreement between the subject subj and the main verb fin by unifying the number and person roles. The final constraint in S-With-Subj's constructional constraints block requires the case of the subject to be nominative.

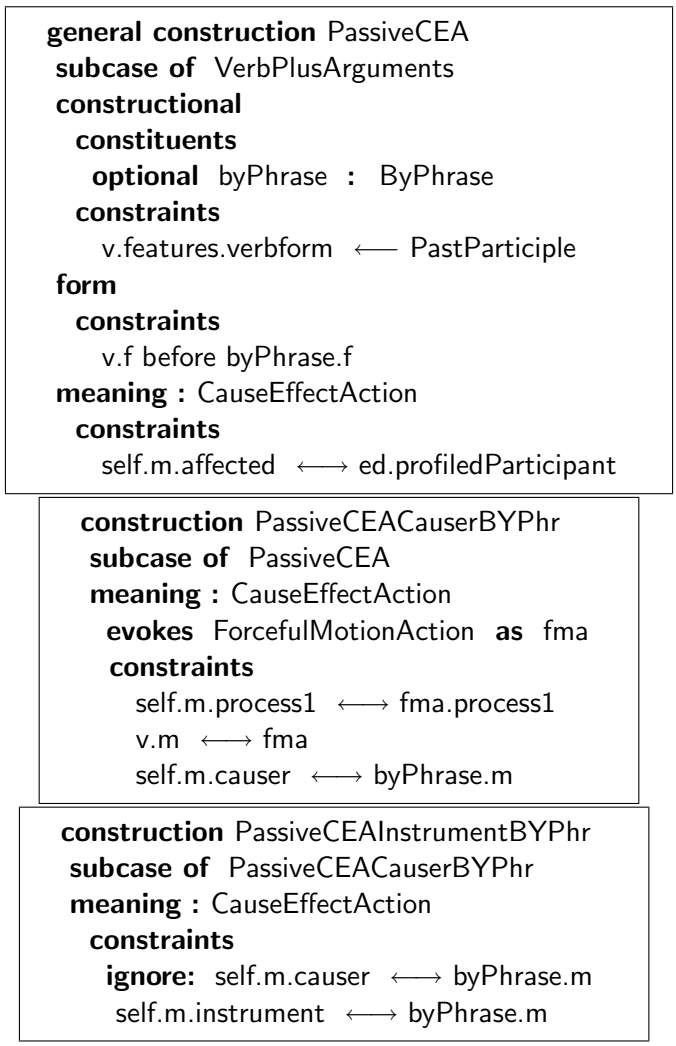

Figure 6.9: The passive versions of the argument structure constructions shown in figure 6.6.

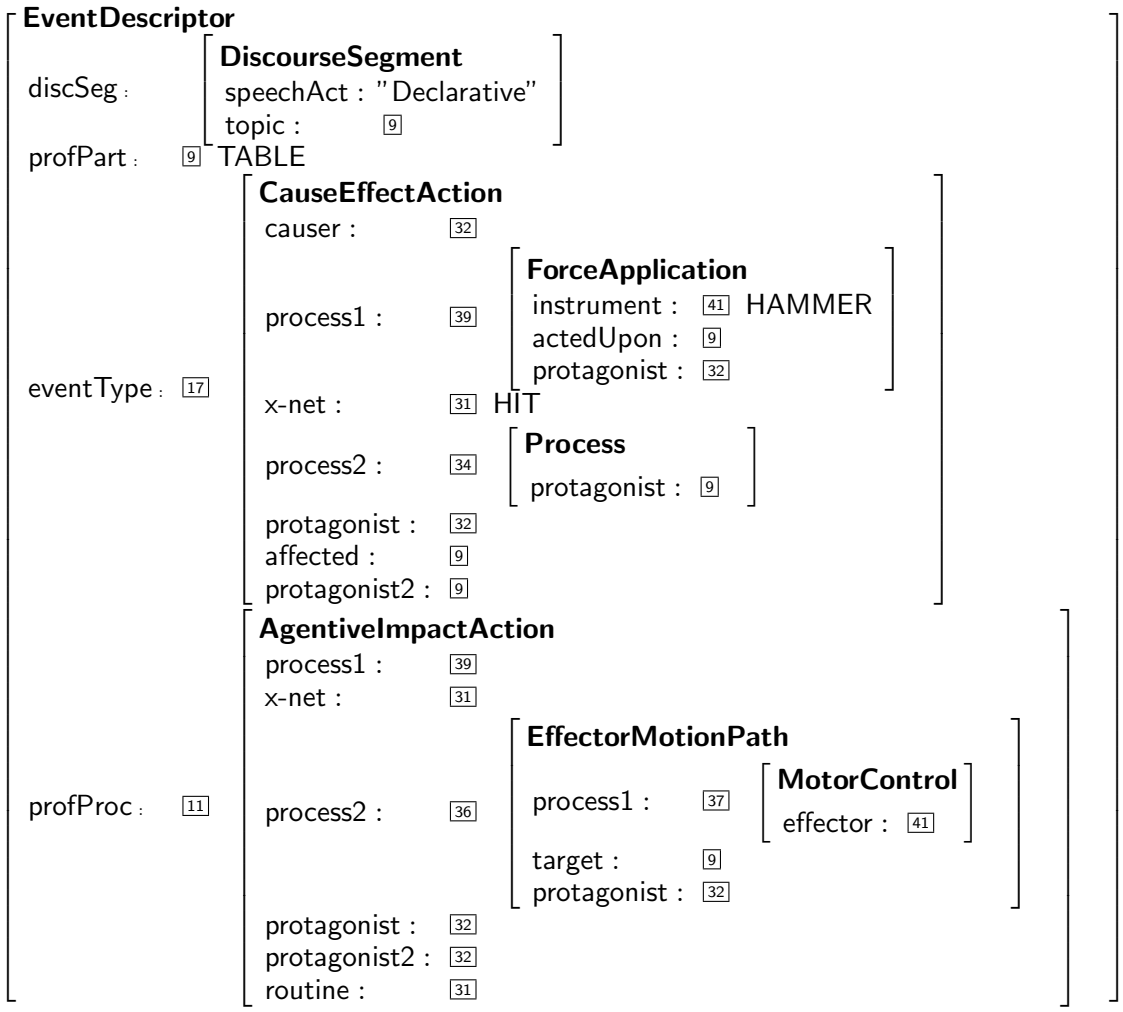

Figure 6.10: The semspec for the sentence, *the table was hit by the hammer*.

Construction S-With-Subj also has an important semantic function. It inherits the EventDescriptor meaning pole type, and adds a constraint to unify the meaning pole of the subject subj constituent with the profiledParticipant role. This constraint thus defines the semantic correlate of the subject to be the profiledParticipant role, and the meaning of the subject is available through this role.

The Declarative construction is a particular kind of S-With-Subj. It inherits all the constraints from S-With-Subj, and adds a type constraint to the fin constituent requiring the fin constituent to be a VP. The inherited constraint from S-With-Subj requires the VP constituent to be finite. On the form side, the Declarative construction requires the subj constituent to come before the fin constituent, enforcing normal declarative constituent ordering. On the meaning side, the Declarative construction unifies the fin constituent's evoked EventDescriptor ed be unified with the its own EventDescriptor. It also sets the DiscourseSegment.speechAct role to "Declarative".

6.2.3 Basic Questions

The semantics of questions have not been fully worked out within the EJ1 grammar. However building on the constructions from the previous section, one can write question constructions with a simplified representation of question semantics. The basic strategy is to subcase the S-With-Subj construction to incorporate subject-auxiliary inversion, and then define yes-no questions and basic WH-questions[6] from there.

Figure 6.12 shows the general S-With-Finite-Aux-NonFinite-AS construction, a construction for yes-no questions (Yes-No-Question), and construction Fronted-WH-Question which represents simple wh-questions. Construction S-With-Finite-Aux-NonFinite-AS represents sentences with a finite auxiliary main verb (constituent fin) and an additional nonfinite argument structure construction (constituent argstruct). Construction Yes-No-Question extends S-With-Finite-Aux-NonFinite-AS, requiring that the fin, subj, and argstruct constituents are in that order. It also sets the DiscourseSegment.speechAct to YN-Question which is a shallow way to show that this construction is a yes-no question.

[6]For a alternative construction-inspired account of questions within the HPSG framework, see Ginzburg and Sag [25] extensive account of questions.

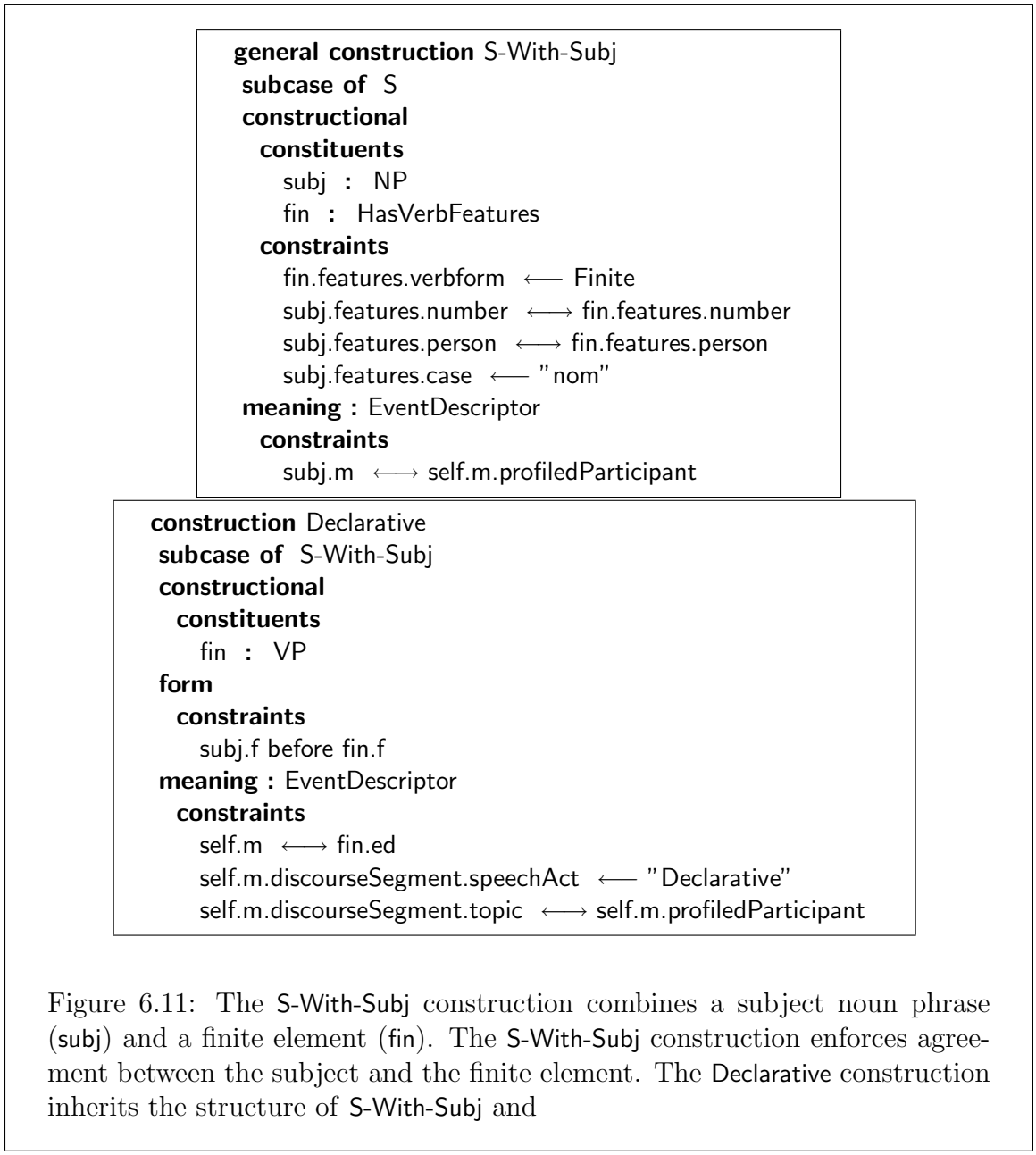

Figure 6.11: The S-With-Subj construction combines a subject noun phrase (subj) and a finite element (fin). The S-With-Subj construction enforces agreement between the subject and the finite element. The Declarative construction inherits the structure of S-With-Subj and

Figure 6.12 shows the question constructions in the EJ1 grammar. Construction Fronted-WH-Question subcases construction Yes-No-Question, inheriting the subj, fin, and argstruct constituents. To these constituents, construction Fronted-WH-Question adds an extraposed constituent qp of type WH-Phr. Constituent qp is intended to cover noun phrases like *who* or *which table* and how-phrases like *how much* or *how far*. Additionally, constituent qp is notable because it is an extraposed constituent. The extraposed keyword tells the analyzer that constituent qp will fill a nonlocal constituent of a construction instantiated later in the derivation. When the analyzer finds this construction, it will bind the semantics of qp into the semantics of the appropriate constituent. Of course, the constituent must have a probability of being nonlocal that is greater than zero.

The form constraints of construction Fronted-WH-Question requires that constituent qp be immediately before the auxiliary fin. In the meaning block of construction Fronted-WH-Question, the meaning of qp is bound to the topic role of the DiscourseSegment schema which indicates that the qp is the topic of the question itself.

Figure 6.13 shows the semspec for the question *which table did he hit*. The semspec in figure 6.13 is almost exactly like the semspec for *he hit the table*, except the DiscourseSegment.topic is not bound to the profiledParticipant role, but rather to the actedUpon role. Additionally, the DiscouseSegment.speechAct role is set to "WH-Question".

6.2.4 Control and Raising

The terms raising and control describe sentences where a frame evoked by a subordinate clause has a semantic argument in a nonlocal, but predictable position (usually subject). Compare:

- He hit the table
- He seemed to hit the table

In both cases, "he" is the hitter, although in the second sentence, the infinitival phrase is subordinate to the raising verb "seemed".[7]

[7]The semantics of raising verbs such as "seemed" has not been worked out.

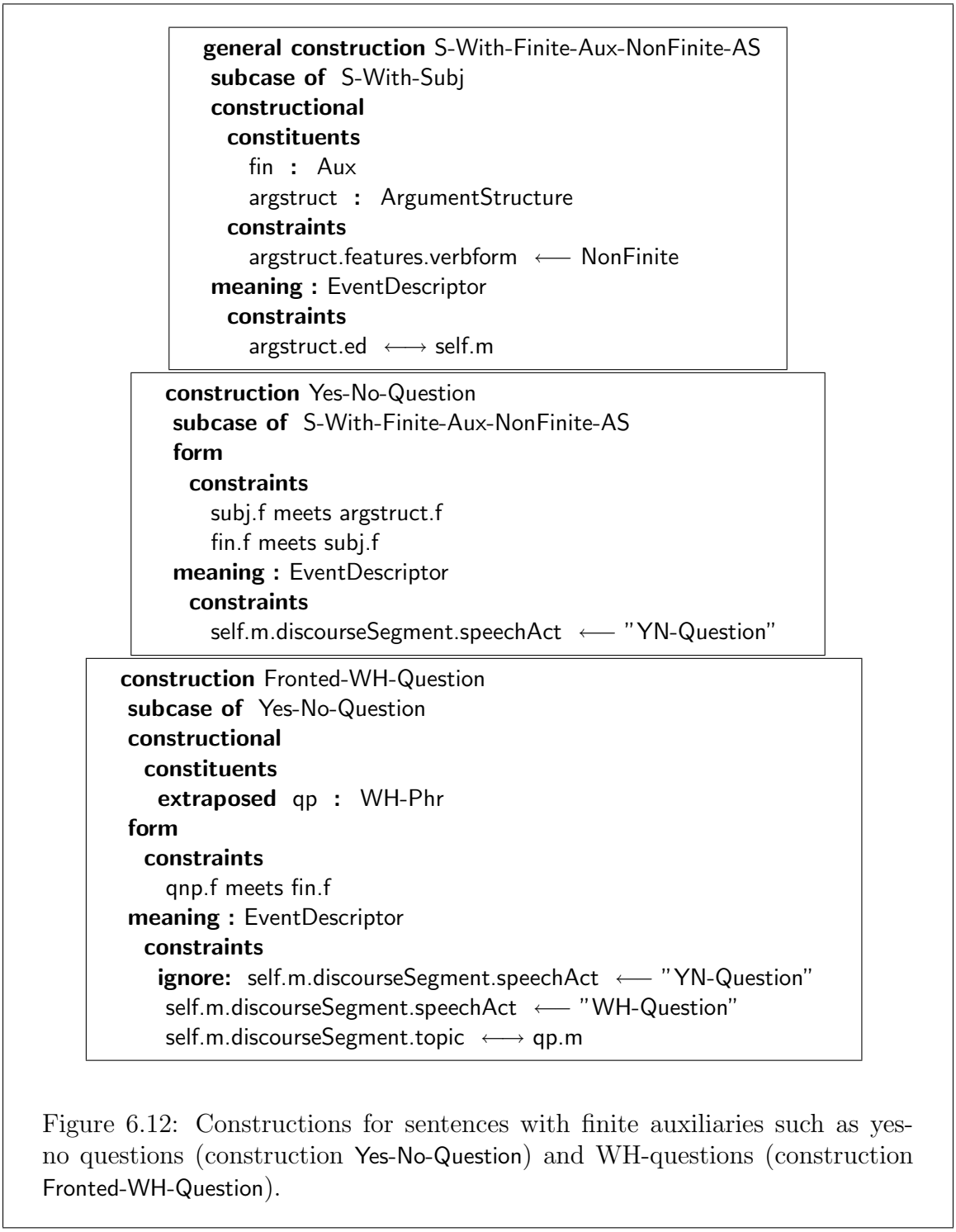

Figure 6.12: Constructions for sentences with finite auxiliaries such as yes-no questions (construction Yes-No-Question) and WH-questions (construction Fronted-WH-Question).

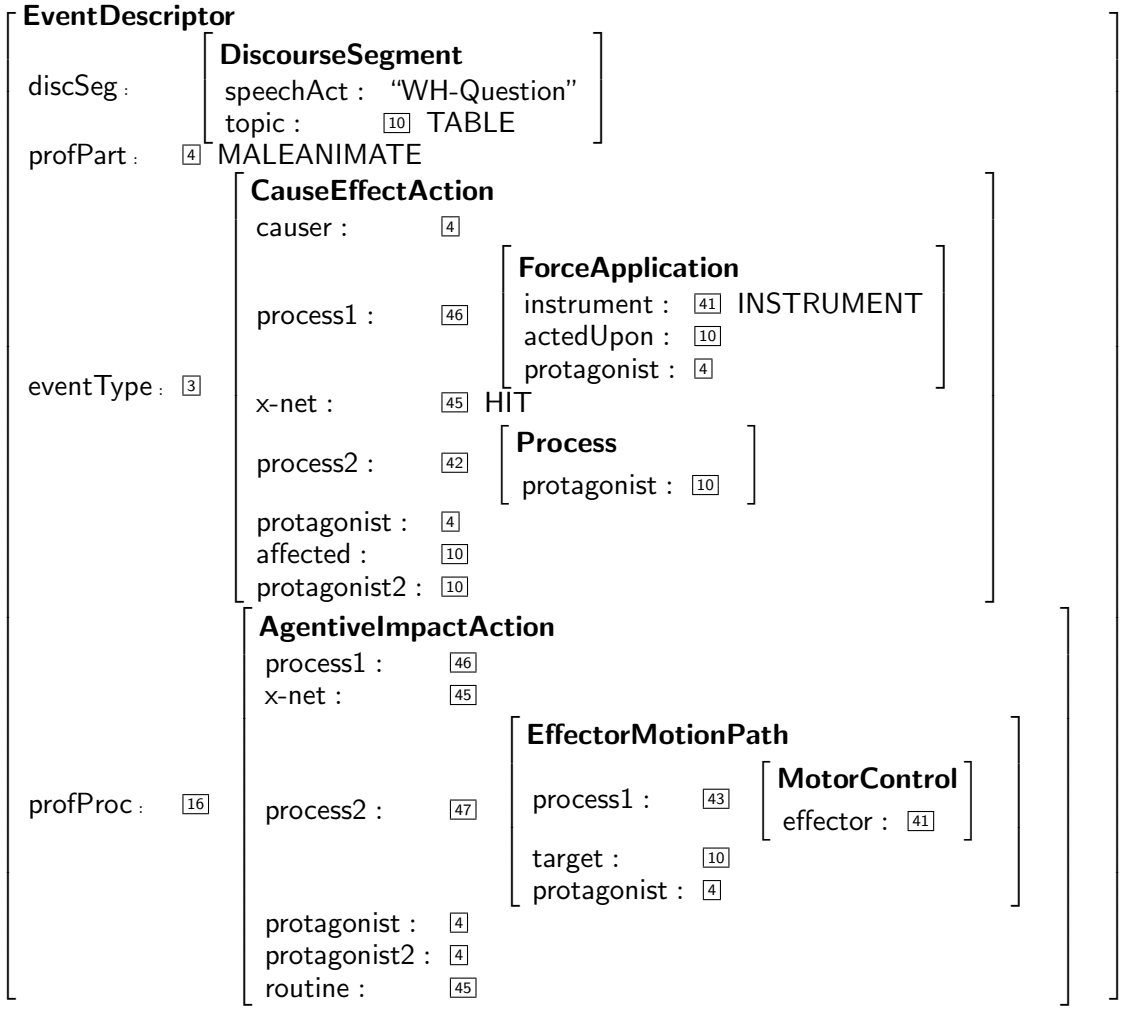

Figure 6.13: The semspec generated by the analyzer for the sentence *Which table did he hit.*

Control is like raising with the additional stipulation that the subject of the sentence is also a meaningful semantic argument of the control verb. Compare:

- He hit the table
- He wants to hit the table

As with the "seemed" case, "he" is the hitter in both sentences. But in *He wants to hit the table*, "he" is also a semantic argument of the control verb "want".

Given what we have seen so far, the obvious technique for dealing with raising and control is to leverage the power of the profiledParticipant role in the EventDescriptor. I will limit my discussion to control cases. Figure 6.14 shows the schematic abstraction over control verbs that I use here. A ControlVerbProcess is a process which has an additional event role. For a sentence like, *I want him to go to the store*, the event role would be filled by the meaning of "him going to the store".

Figure 6.14 also shows an argument structure construction for subject control cases like, *wants to hit the table*. Out of the constructions presented in this chapter, the SubjectControl construction has a unique property: It is an argument structure construction which takes another argument structure construction argstruct as a constituent. The SubjectControl construction also adds infinitive marker to: TO-IM and a control verb v[8] as constituents. Additionally, the constraint in the constructional block requires that the verb in the argstruct constituent be infinitival.

While the form block does the obvious ordering, the meaning block has two interesting constraints. As is the case with most active voice sentences, the first constraint co-indexes protagonist role of the SubjectControl's meaning pole to the EventDescriptor.profiledParticipant role, thus ensuring that the subject of the sentence will be the fill the *want-er* role (for example). The second constraint unifies the EventDescriptor.profiledParticipant role in the argument structure constituent argstruct with the SubjectControl's EventDescriptor.profiledParticipant. Assuming that the subordinate argument structure construction argstruct has co-indexed the appropriate semantic role to its profiledParticipant role, this constraint guarantees that the subject will also fill the appropriate semantic role in argstruct.

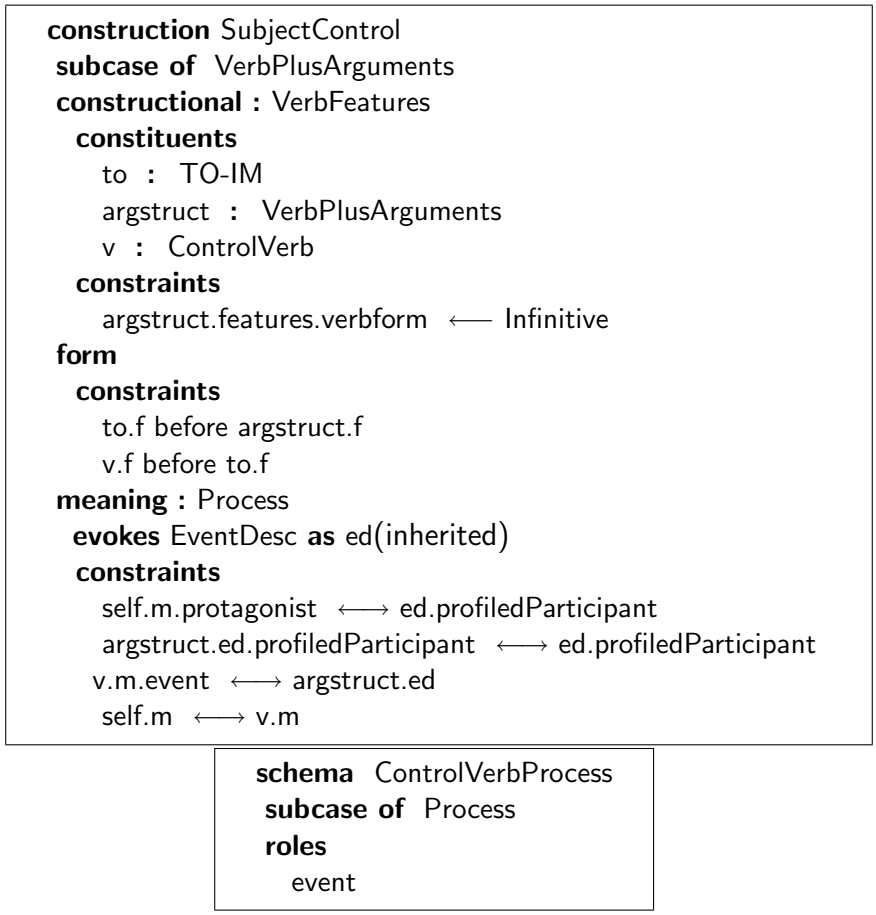

Figure 6.14: A construction for subject control.

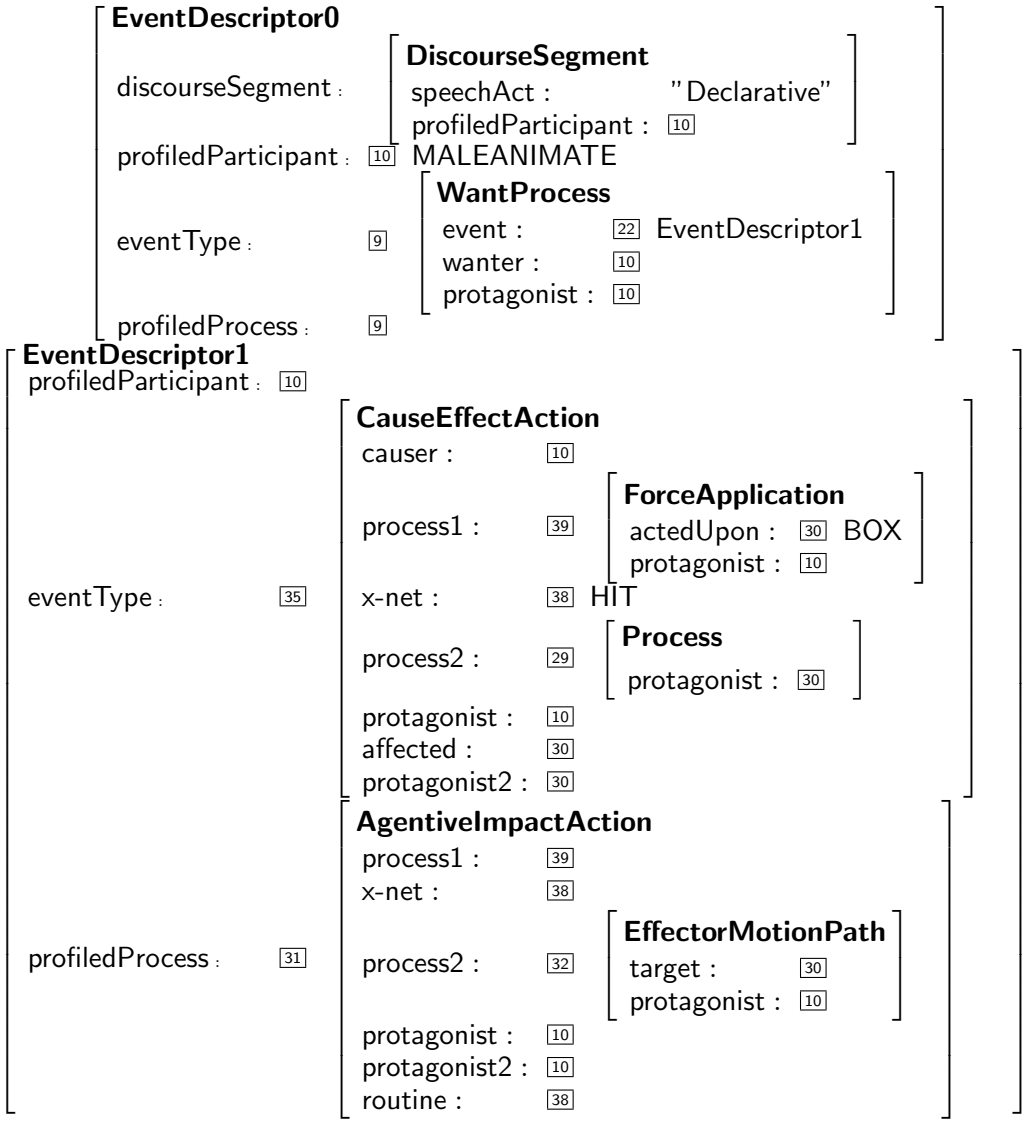

Figure 6.15: A semspec for *he wanted to hit the box*

```
construction ObjectControl
 subcase of SubjectControl
 constructional : VerbFeatures
   constituents
     np : NP
   constraints
     np.features.case ⟵ "acc"
 form
   constraints
     np.f before to.f
     v.f before np.f
 meaning : Process
   constraints
    ignore: argstruct.ed.profiledParticipant ⟷ ed.profiledParticipant
     np.m ⟷ argstruct.ed.profiledParticipant
```

Figure 6.16: A argument structure construction for VP phrases like *wants him to hit the ball*.

Figure 6.16 shows a construction covering a sentence like, *She wants him to hit the ball* which is a case of control in which the object (instead of the subject) of "want" is the agent of hit. For lack of a better name, I am calling this construction ObjectControl. The ObjectControl construction subcases SubjectControl, adding an accusative NP constituent as an object. In its meaning block, the ObjectControl construction ignores the inherited constraint that binds together the profiledParticipant roles, and instead unifies the meaning of the NP with the argstruct.EventDescriptor.profiledParticipant role. This constraint links the object of ObjectControl to the appropriate semantic role in argstruct.

6.3 Ditransitive as a Radial Category in EJ1

The ditransitive construction is highly productive, semantically rich, and even has unpredictable (but motivated) usage such as the creation and obtain examples.

[8] A control verb has a kind of ControlVerbProcess as its meaning pole.

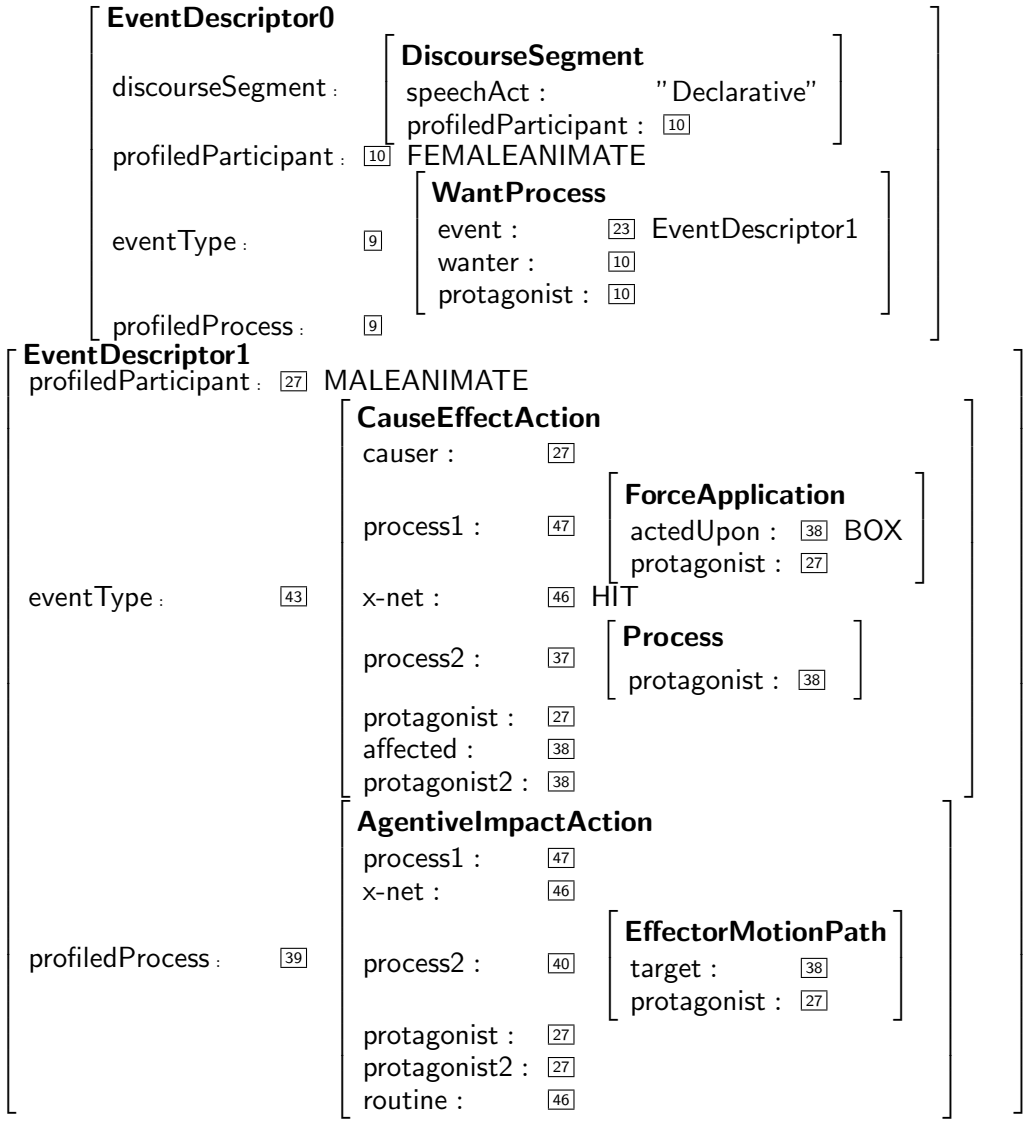

Figure 6.17: The semspec for *she wanted him to hit the box*

Consequently, the ditransitive construction is a great testing ground for theories of construction grammar. Linguists tend to describe it in terms of the verbs that the ditransitive construction can take as a constituent:

- He gave/handed me a book (central transfer)
- He tossed/kicked me a book (cause-motion to transfer)
- He brought/took me a book (joint-motion to transfer)
- He promised/owed me a book (responsibility to transfer)
- He denied/refused me a book (deny transfer)
- He bequeathed/willed me a book (future transfer)
- He made/wrote me a book (creation with intent to transfer)
- He won/stole me a book (obtain with intent to transfer)
- He permitted/allowed me a book (enable transfer)

Goldberg [26] has studied the ditransitive construction extensively and has provided a radial category structure for the different verb senses that are compatible with the ditransitive. While her work is extremely insightful, she does not precisely specify how the meaning of the verb and the meaning of the construction relate, nor does she describe her constructions in a way that is compositional with the rest of the grammar.

The section provides an alternative (partial) category structure for the ditransitive construction. Goldberg's work is used as a starting point. The category structure is specified in EJ1, and is compatible with the active and raising constructions described in previous sections. Though the categorization provided here is an improvement over Goldberg's work, it is just a proof-of-concept starting point for future research in specifying the ditransitive within ECG. It is not intended as a definitive representation of the semantics of ditransitive scenes, but these constructions do illustrate the power of deep semantics and best-fit analysis for studying complex linguistic phenomena.

The constructions described in this section are arranged in a radial structure with the central transfer sense being the root of the ditransitive hierarchy. Constraint overrides (through ignore) are used to redefine the meanings associated with the verb of the radial extensions, and thus the inheritance is a principled partial inheritance. This is consistent with how Goldberg construes the inheritance relations between the different senses of the ditransitive.

6.3.1 Central Ditransitive

Figure 6.18 shows three schemas for representing the central literal meaning of the ditransitive argument structure construction. Schema ObjectTransfer is a complex process combining two different actions. process1 is a ReleaseHold action and process2 is an EstablishHold schema. Both ReleaseHold and EstablishHold are kinds of ForceApplication which means that they have semantic roles for a actor and an actedUpon participant. ObjectTransfer also adds roles for a getter, a giver, and a theme. The constraints block of ObjectTransfer binds the primary protagonist to the giver role, the secondary protagonist2 role to the getter, and additionally co-indexes both of the actedUpon roles with the theme. Thus this schema defines a scene in which the giver releases a hold on the theme, while the getter establishes a hold on the theme.

Figure 6.19 shows the CentralActiveDitransitive construction and a lexical GivePastTense construction. The CentralActiveDitransitive is an active form VerbPlusArguments that inherits a verb constituent v and an evoked EventDescriptor and defines two NP constituents np1 and np2. The form block constrains the ordering to be v, then np1, np2.[9] The meaning bloc of CentralActiveDitransitive sets the theme role to the meaning of np2, the getter role to the meaning of np1, and the giver role to the profiledParticipant role of the evoked EventDescriptor. Importantly, CentralActiveDitransitive unifies the meaning of the verb v.m with the meaning of the construction. This requires that any verb fitting this construction must denote an ObjectTransfer schema in its meaning pole.

[9]Constituent heaviness leading to alternative constituent arrangements will not be addressed here.

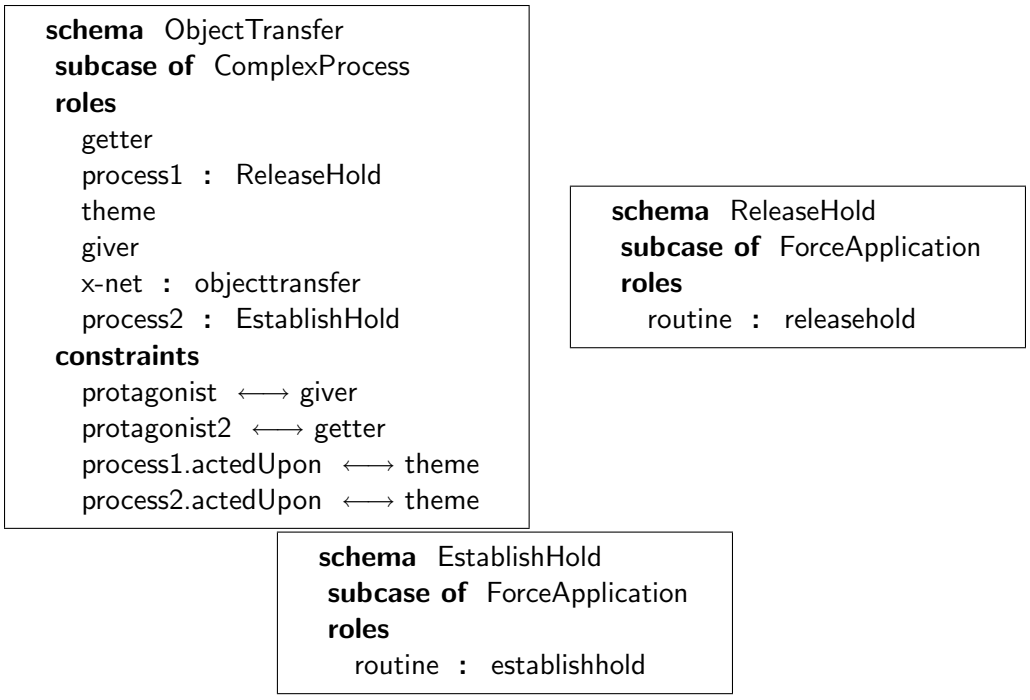

Figure 6.18: Simplified schemas for representing Transfer scenes.

The GivePastTense construction shows an example lexical "gave" construction. Like the lexical SlidePastTense construction, the construction sets its meaning to the frame in which it is defined, and then it sets its x-net role to the appropriate x-net. Because lexical *give* specifies that its meaning is an ObjectTransfer schema, it can be bound to the verb constituent of CentralActiveDitransitive.

The semspec shown in figure 6.20 shows how the verb's meaning and the meaning of CentralActiveDitransitive support inference for the sentence *she gave him a cookie*. In figure 6.20, the profiledParticipant is FEMALEANIMATE (standing in for "she"), the eventType and profiledProcess is an ObjectTransfer. The ObjectTransfer schema has the getter set to MALEANIMATE the giver is set to FEMALEANIMATE, and the theme set to the COOKIE. The COOKIE is also the actedUpon entity for both the ReleaseHold and the EstablishHold schema instances.

6.3.2 Cause-Motion Ditransitive

The ActiveDitransitiveCauseMotion construction shown in figure 6.21 is the ditransitive extension for cause motion verbs like *toss* and *throw*. The ActiveDitransitive-CauseMotion construction is a subcase of CentralActiveDitransitive. Thus it inherits the constituents and form constraints from CentralActiveDitransitive. Importantly, it inherits its meaning type of ObjectTransfer from CentralActiveDitransitive. This defines the semantic similarity between *She gave him a cookie* and *She threw him a cookie*.

The meaning block is where ActiveDitransitiveCauseMotion distinguishes itself from CentralActiveDitransitive. ActiveDitransitiveCauseMotion evokes a CauseMotionPathAction schema. The CauseMotionPathAction is a special kind of CauseEffectAction in which the result process process2 is a MotionAlongAPath. The constraints block in ActiveDitransitiveCauseMotion overrides the inherited constraint that requires that the verb's meaning v.m is bound to the meaning of construction as a whole. Instead of unifying the verb's meaning to the construction's meaning, the second constraint requires that the verb's meaning is unified with the evoked CauseMotionPathAction. This constraint requires that the verbs used with this construction are cause-motion verbs. The final two constraints in the meaning block hook up the CauseMotionPathAction with the

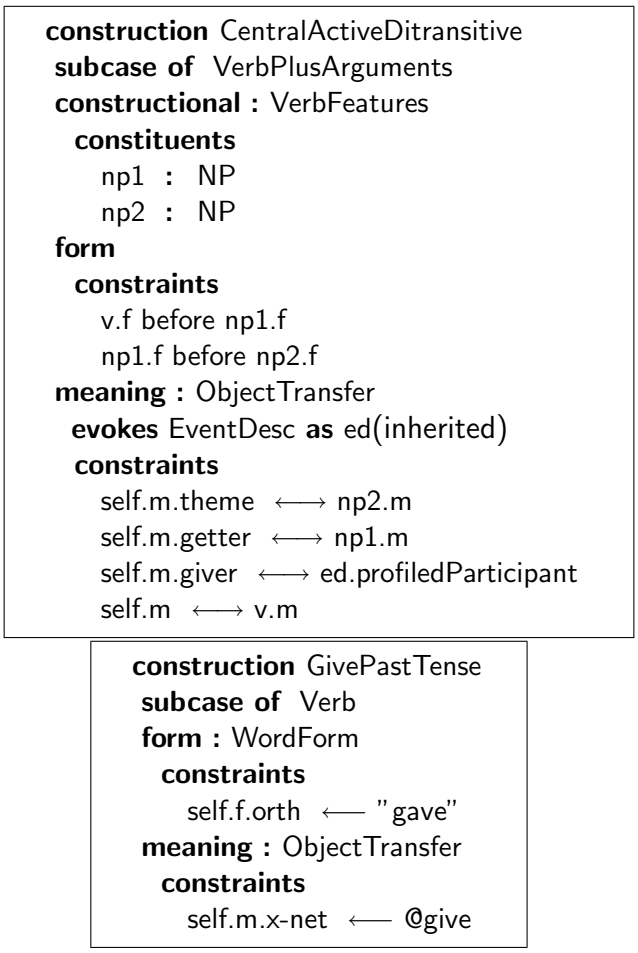

Figure 6.19: The central ditransitive construction and an example "gave" construction

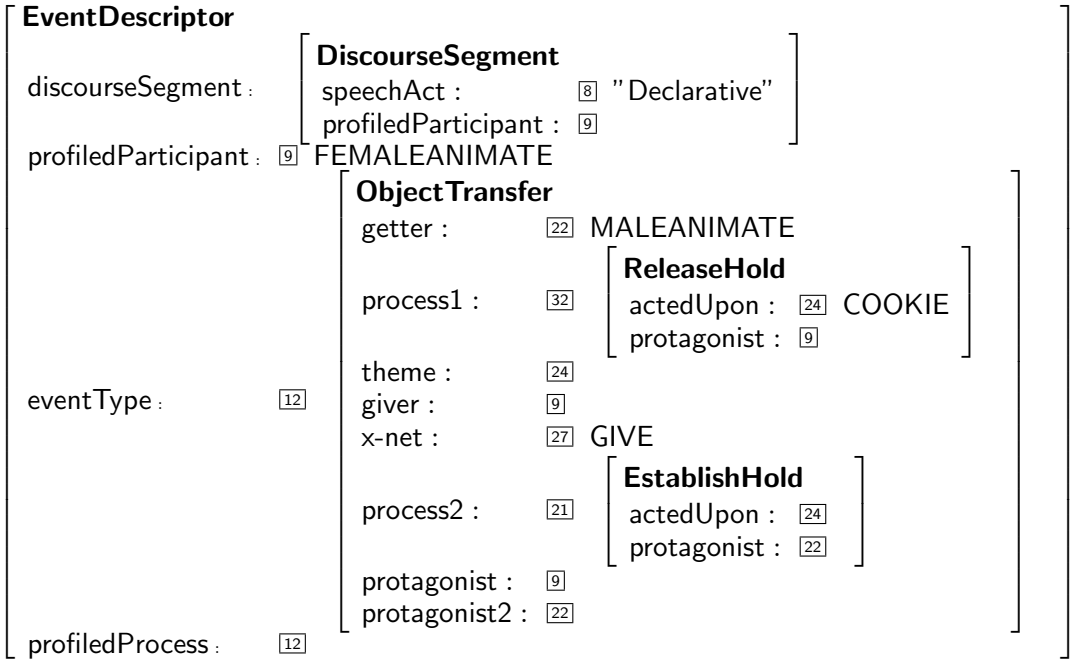

Figure 6.20: The semspec for the sentence, *she gave him a cookie.*

ObjectTransfer schema, unifying the process1 roles of the two schemas and co-indexing the getter role with the spg.goal role in the CauseMotionPathAction.

The ThrowPastTense construction also in figure 6.21 sets its meaning to the CauseMotionPathAction schema. The ThrowPastTense construction shows the power of the x-net role because the detailed motor control information that describes the differences between a *throw*, a *toss* and a kick are built into the x-net role, allowing a general schema like CauseMotionPathAction to act as the interface with the argument structure constructions.

Figure 6.22 shows the semspec for *she threw him a cookie.* The ObjectTransfer schema instance specifies the appropriate bindings: the getter set to MALEANIMATE the giver is set to FEMALEANIMATE, and the theme set to the COOKIE. The COOKIE is also the actedUpon entity for both the ReleaseHold and the EstablishHold schema instances.

An additional CauseMotionPathAction schema instance is also in the semspec. This corresponds to the two construals of a cause-motion ditransitive as both a cause-

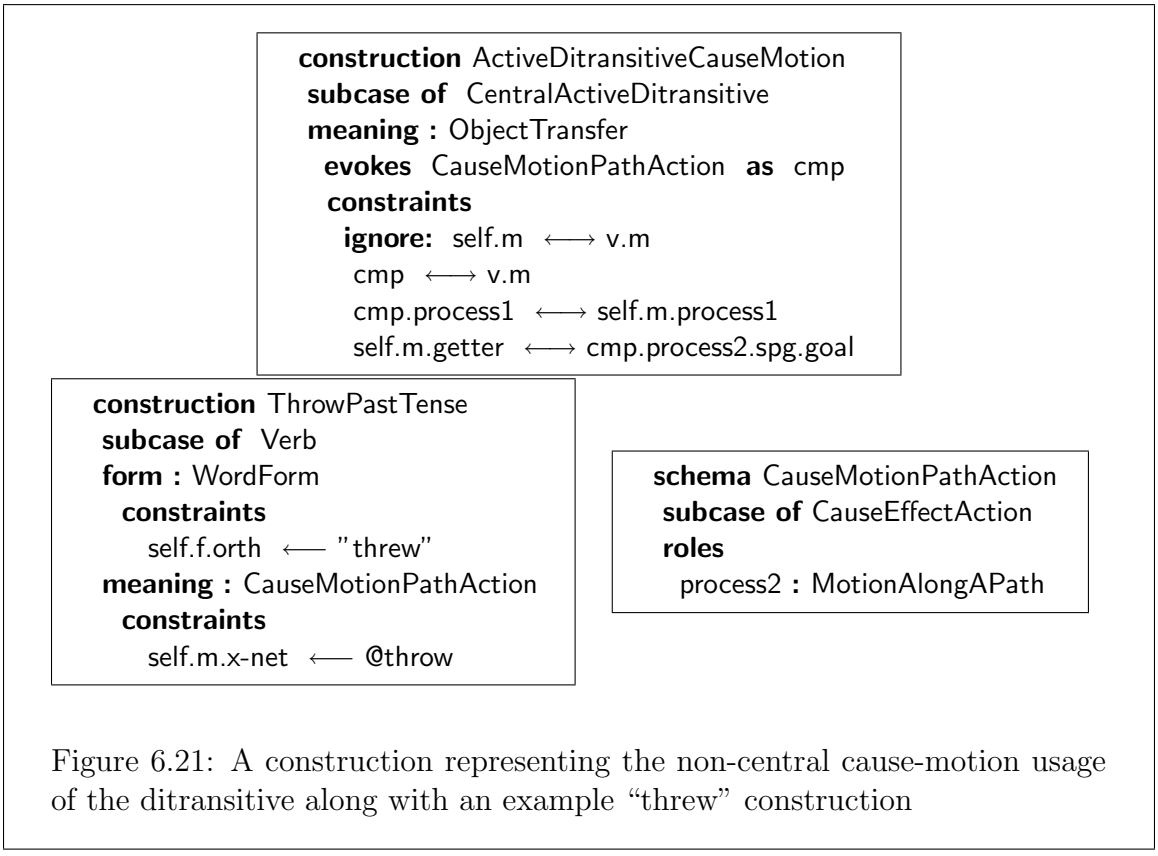

Figure 6.21: A construction representing the non-central cause-motion usage of the ditransitive along with an example "threw" construction

motion and a transfer. The causer role of the CauseMotionPathAction is bound to FEMALEANIMATE, and the COOKIE is set to the affected role. In this construal of the scene, the goal role of the MotionAlongAPath's SPG is bound to MALEANIMATE.

An additional inference made in this formulation is that the ReleaseHold of the ObjectTransfer is unified with process1 of the CauseMotionPathAction. While one could justifiably argue about whether this is the best formalization, the point here is that the construction takes the two scenes and creates shared structure through unification. The end result being two construals of the same action with shared sub-structure.

Importantly, the profiledProcess and eventType roles in the EventDescriptor are bound to different schema instances. The profiledProcess is bound to the CauseMotionPathAction because of *threw*, and the eventType role is bound to the ObjectTransfer because of the argument structure construction. In central case constructions, the eventType and the profiledProcess are unified, but non central argument structure constructions retain the eventType, but profile a different, but related process.

6.3.3 Creation Ditransitive

The ditransitive construction that takes creation verbs is an extremely interesting construction. It combines a transitive verb that has nothing directly to do with transfer, and treats the created thing as the theme for an intended, but still hypothetical transfer. This creation ditransitive is a great example of how the meaning of a construction can be a motivated, but unpredictable semantic composition of its parts.

Figure 6.23 shows schemas and constructions used to represent the creation extension of the ditransitive. The Intention schema is a simplified representation of an actor hoping to perform some intendedAct, and the CreationAction schema is a simplified representation of a Process in which a creator creates a createdThing.

Figure 6.23 also shows a ActiveCreationDitransitive construction and a lexical BakePastTense construction. The ActiveCreationDitransitive inherits the form and constructional constraints from CentralActiveDitransitive, but is semantically quite different. It evokes a CreationAction as ca and an Intention as int. It overrides the inherited constraint

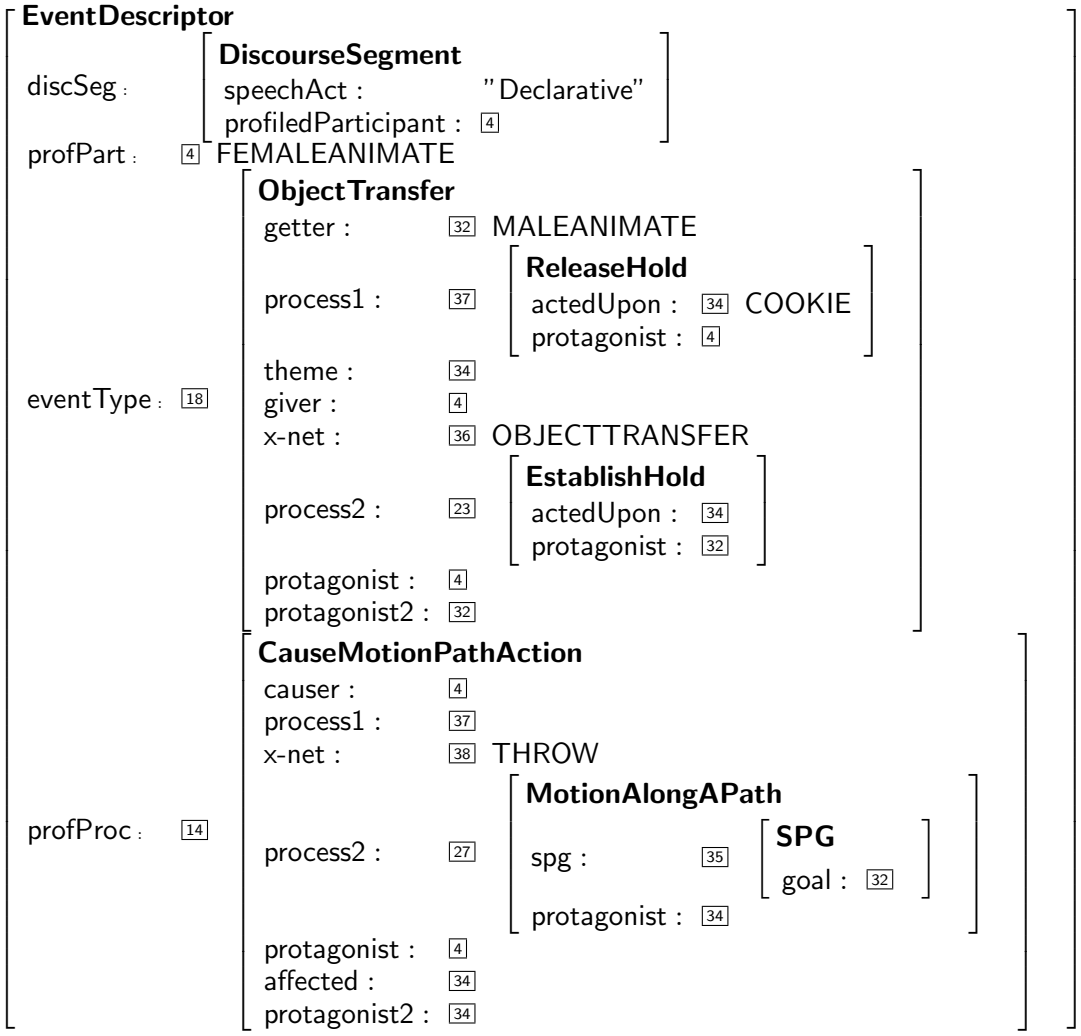

Figure 6.22: The semspec for the sentence, *she threw him a cookie*.

binding the meaning of the verb to the ObjectTransfer type on the meaning pole, and instead unifies the meaning of the verb with the CreationAction. This constraint specifies that only CreationAction verbs are semantically compatible with this construction. The next two constraints link the roles of the CreationAction with the ObjectTransfer, unifying the theme with the createdThing and the creator with the giver. The last two constraints unify the roles the Intention schema with the rest of the scene, co-indexing the creator role with the Intention.actor role, and binding the complete ObjectTransfer to the intendedAct role.

The lexical BakePastTense construction follows the same pattern for lexical constructions that we have seen throughout this chapter. It specifies the lexical form, subclassing Verb. On the meaning side, it defines its meaning block to the frame it belongs to, and then binds the lexically specific x-net information into the x-net role.

The semspec in figure 6.24 has the same basic properties as the semspec for *she threw him a cookie* with the notable exception of a CreationAction filling the profiledProcess role, and an additional Intention schema instance. Roles profiledParticipant, giver, and creator, Intention.actor are all bound to FEMALEANIMATE. Roles theme and createdThing are bound to the COOKIE. As usual, MALEANIMATE is bound to the getter role. Additionally, the Intention.intendedAct is bound to the ObjectTransfer itself.

6.4 The Tip of the Iceberg

Even though the constructions and semspecs in this chapter are fairly complex, these results are just the tip of the iceberg. In addition to demonstrating the capabilities of the system, the goal of this chapter is to show future grammar researchers examples of the formalism and analyzer at work. Hopefully these examples will help point them in the right direction.

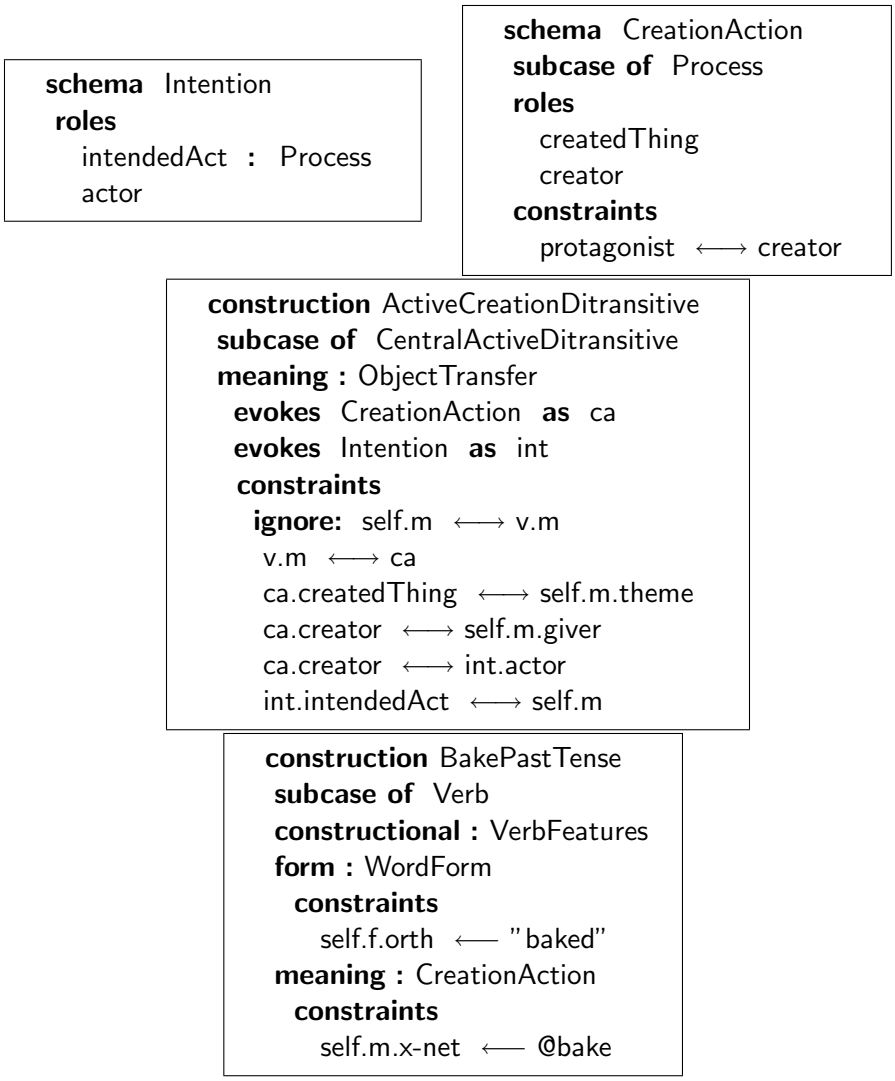

Figure 6.23: The creation sense of the ditransitive construction covering sentences like *He baked her a cookie*. An example Bake construction is shown along with simple Intention and CreationAction schemas.

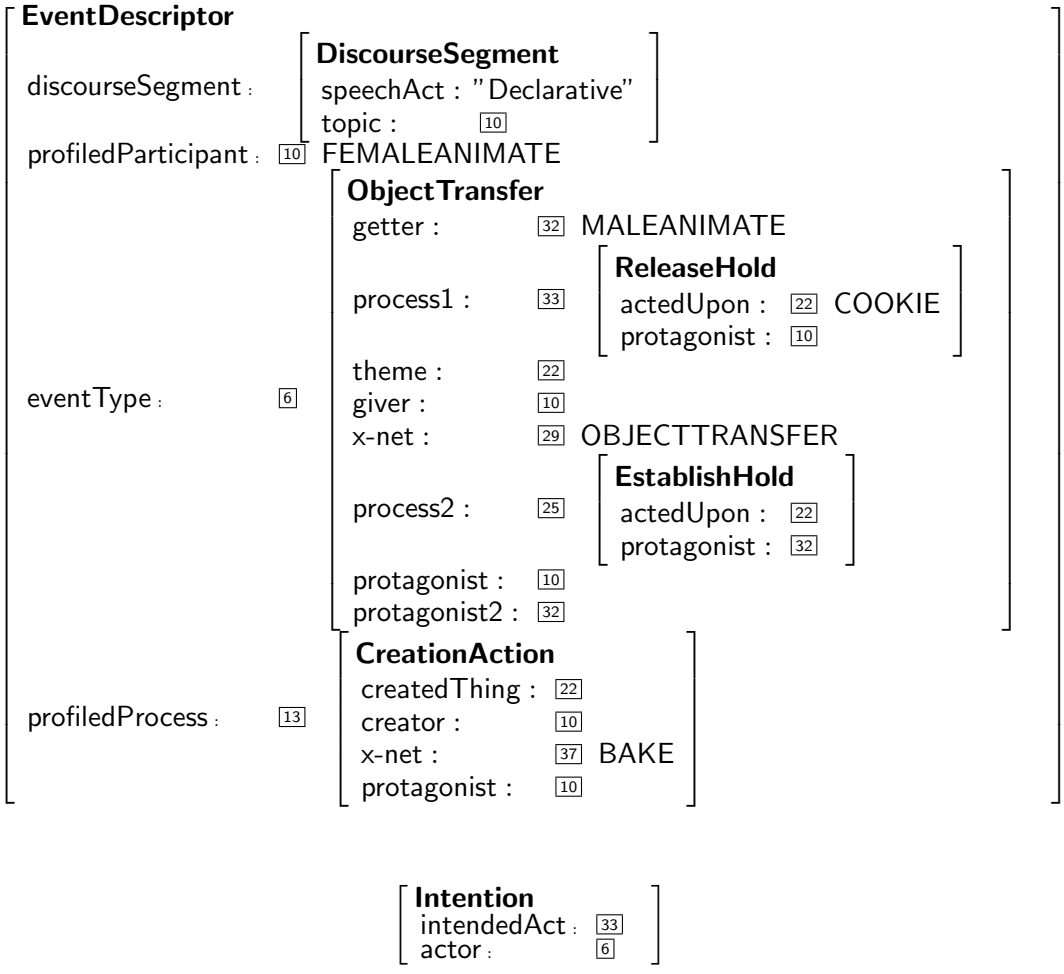

Figure 6.24: The semspec for the sentence, *she baked him a cookie.*

Chapter 7

Predicting Reading Time

The constructional analyzer is the first model of deep semantic interpretation to make detailed reading time predictions. This chapter shows how the constructional analyzer can be used to model the reduced-relative/main-verb ambiguity data produced by McRae, Spivey-Knowlton and Tannenhaus [51]. As far as I know, the reading time predictions are the best modeling results of the McRae et al. data.

The goal of using a construction-based model of interpretation for psycholinguistic research goes back to Jurafsky [34]. He observes that in construction grammar, lexical items, idioms, and general syntactic-semantic argument linking rules are represented by constructions. This uniformity of representation makes a construction-based model capable of making predictions about a wide-range of linguistic phenomena. Thus the fact that the analyzer uses constructions is an important feature of the system because it allows the system to be compatible with modeling many different kinds of phenomena.[1]

This rest of this chapter is structured as follows. The motivation for using a model of interpretation to predict reading time and related work are covered in section 7.1. More on the data and its relevance to models of interpretation are covered in section 7.2. Section 7.3 shows how I define the constructions and extract the necessary parameters for modeling the McRae et al data. Section 7.4 precisely defines the link between the analyzer's internal probabilistic state and predictions about

[1] Though the modeling of such phenomena will be left to future work.

processing difficulty. Section 7.5 shows a word-by-word trace of the system on the two experimental conditions investigated by McRae et al. Section 7.6 shows how the predictions of the system compare to the experimental data. how I link the internal state of the analyzer to reading time predictions.

7.1 Motivation and Related Work

Like the constructional analyzer, Jurafsky's model [34] is a probabilistic, cognitively plausible model of interpretation that uses constructions. He uses a general probabilistic parsing algorithm to parse utterances, and defines various operating principles for determining when the reading of a sentence should be difficult. However his model cannot make detailed incremental reading time predictions.

There are three other psychologically plausible models that can predict the McRae et al. data. They include:

- Pado [59] uses the broad-coverage syntactic parser designed by Roark [68] in combination with a separate frame-based semantic model to make detailed reading time predictions on a range of linguistic data including McRae et al. Pado assumes reading time difficulty whenever the syntactic parser and her semantic model disagree about which parse should be the top ranked parse. She correlates the amount of disagreement with reading time difficulty. However Pado's model cannot do deep semantic interpretation, and it does not use constructions.

- The model designed by Narayanan and Jurafsky [57] is a small scale construction-based model that uses a factored probabilistic model over syntax and semantics. The factored model is implemented within a graphical model in which support for the competing interpretations is computed using a noisy-AND. The graphical modeling framework enables Narayanan and Jurafsky to precisely define the operating principles defined by Jurafsky [34]. Crucially, Narayanan and Jurafsky use the probabilistic state of the graphical model to model a fragment of the McRae et al. data.

- The original McRae et al. article also presents models of their data, but the computational models they define are ad hoc collections of parameters that interact in a connectionist framework. In their models, each interpretation competes for activation with the competition's duration corresponding to reading time difficulty.

Given this range of models that already predict the McRae et al. data, why should it matter that the constructional analyzer can model the same data? A psychologist might argue that if two models both predict the data equally well, the simpler model is a better model. Given that the other models are all simpler should Occam's razor make the constructional analyzer unnecessary?

While I agree that simplicity is a virtue, it is not the only criteria for designing a cognitive model. A cognitive model should also be compatible with related findings from cognitive science. Such a model has insight even if its predictions can be made by a simpler model.

It is obvious that speakers of a language do more than just modulate their processing time when they read a sentence. If they choose to read the sentence at all, then their goal is to understand what it means. Thus a single model like the constructional analyzer that can predict reading time difficulty *and* model the understanding of the utterance is more simple than a theory that requires one model for predicting reading time and another for understanding.

But getting caught up in arguments about simplicity is missing the point. The constructional analyzer was designed to be a cognitively plausible model of interpretation, not a program to predict reading time. Predicting reading time data is only important in that it provides another way to suggest that the constructional analyzer is indeed cognitively plausible.

7.2 Experimental Data

Consider the following pair of sentences.

1. The cop arrested by the detective was guilty.

2. The crook arrested by the detective was guilty.

Both of these sentences use the reduced relative construction (arrested by the detective) construction, but importantly, the subjects differ in terms of whether they are a good agent of arrested (e.g. cop) or a good patient (e.g. crook). Importantly, *arrested* is a verb that is ambiguous between a past-participle reading and past-tense reading.

McRae, Spivey-Knowlton and Tannenhaus [51] tested pairs of sentences like the pair above within a self-paced reading paradigm. They found that the word by word reading times for the two classes of sentences differ (see table 7.1) depending on whether the sentence had a good agent subject or a good patient subject. Broadly speaking, sentence one, while initially easier for readers to process at the word *arrested*, is harder for readers to process in the range of *by the detective was*.

The basic explanation for the reading time difference provided by [51], [56] and [60] is that reading time differences are a consequence of violation of semantic expectation: Because *arrested* is ambiguous between the past tense and the past participle and cops are prototypical arresters, the expected interpretation of *the cop arrested* is highly biased towards an interpretation in which the cop is doing the arresting. When the phrase *by the detective* is encountered, the expectation that the cop is doing the arresting is violated, leading to a higher reading time over the baseline non-reduced relative version of the sentence.

In sentence two, the scenario is reversed. The best purely syntactic analysis of the sentence prefix, *the crook arrested* is one in which *arrested* is the main verb and the crook is doing the arresting. But this is at odds with our notion that thieves tend to be arrested, and not the other way round. Upon encountering, *by the detective*, the reader then gains syntactic evidence for the reduced relative interpretation, and the disagreement is resolved, which leads to less of a delay at *by the detective was* over baseline than in the *cop arrested* case.

Importantly, the McRae et al. results show that differences in semantics affect the word-by-word interpretation of a sentence. Further, it brings into question any hypothesis maintaining that syntax is strongly autonomous. And finally, it shows that

sentence position	Good Agent	Good Patient
X-ed by	24 ms	61 ms
the Agent	42 ms	32 ms
was ...	44 ms	-1 ms

Table 7.1: The average reading time delay over the unreduced relative baseline reported by McRae et al [51]. These results are averaged over 40 sentence pairs. The self-paced reading time experiment had a sliding window showing two words at a time. First the subject was shown, then the ambiguous past participle + by (e.g. "arrested by"), then the actual agent (e.g. "the detective").

a cognitively plausible model of human sentence processing must include a model of semantics, and that the semantic model needs to condition processing choices at an incremental level.

7.3 Constructions and Parameters

Before the constructional analyzer can be used to model the data from McRae et al., one must write a grammar for the test sentences and define the relevant syntactic and semantic parameters for the factored model.

7.3.1 The Grammatical Rules

To define the lexical constructions, I ran each sentence from McRae et al. through a part of speech tagger, and built small lexical hierarchies rooted by a general part of speech construction. The phrasal and sentential constructions are hand written for this task, however they are consistent with the constructions defined in chapter 6.

Consider the lexical constructions for *arrested* shown in figure 7.1. VBN is the Penn part of speech tag for past participles, and so I used general construction VBN as the root of the past participle hierarchy. Arrested-VBN is the corresponding past participle lexical construction for *arrested*. VBD is the Penn tag for past tense verbs,

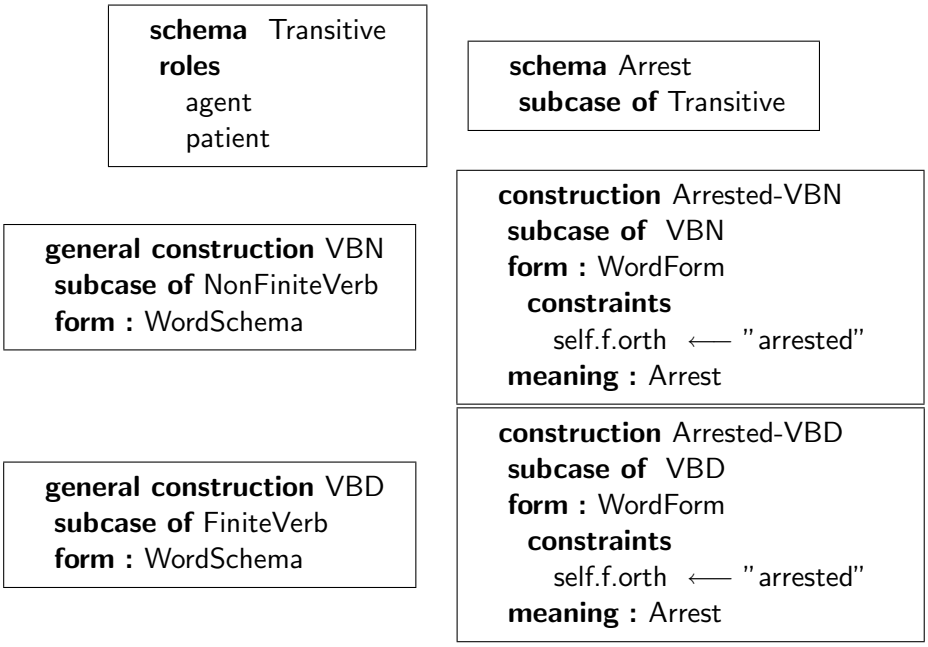

Figure 7.1: The competing lexical constructions for the word *arrested* as well as a schema representing the meaning of Arrest using simple thematic roles.

and thus there is construction Arrested-VBD which is the corresponding lexical construction for past tense *arrested*. Both constructions specify that schema Arrest is the simple (transitive) meaning of the verb arrest. Obviously, in a real grammar, the semantics would be more like the schemas defined in chapter 6, but for the limited purpose of predicting reading time, this simple notion of semantics is satisfactory.

Three of the important phrasal constructions for the task are shown in figure 7.2 and figure 7.3. Construction ActiveTransitiveVP is a simple transitive VP construction. It has constituents for a verb v, an object patientNP, and two optional PP modifiers lightMod and mod. The form block constrains the ordering to be v, lightMod, patientNP, mod. The meaning block evokes a Transitive schema and binds it to both the meaning of the construction and the meaning of the verb. This limits the verbs that can bind to this construction to just those that are semantically transitive. Additionally, the patient role in the Transitive schema is identified with the meaning of patientNP. The commented probabilities in the constituents block indicate the likelihood of a

constituent being expressed, and if expressed, local. For the patientNP constituent, the likelihood of it being expressed is estimated to be .979, and in this grammar it is constrained to be local with weight 1.0. The two optional constituents just have their probability of being expressed since they cannot be expressed nonlocally. The method for estimating these probabilities will be described below.

Also shown in figure 7.2 is the reduced relative construction NPplusRR which joins an NP constituent k with an ExtractedNonFinitePassiveVP constituent named pvp. The ExtractedNonFinitePassiveVP is the relative clause. The meaning block of NPplusRR identifies the meaning of the construction with the meaning of k and identifies the meaning of k with the patient role of the ExtractedNonFinitePassiveVP constituent.

The construction shown in figure 7.3 is the ExtractedNonFinitePassiveVP construction. Construction ExtractedNonFinitePassiveVP is a kind of NonFiniteVP that has a VBN constituent, an optional ByPhrase constituent, and an optional PP modifier. The form block requires that the verb come before the ByPhrase and that the ByPhrase comes before the modifier mod. The meaning block is typed to be the Transitive schema shown in figure 7.1. The meaning of the ByPhrase is bound to the agent role, the meaning of the construction itself is bound to the meaning of the verb which requires that only verbs with Transitive semantics be constituents of this construction.

7.3.2 Estimating Parameters

The constructional analyzer uses three kinds of parameters: the constituent specific locality parameters, the constituent filler parameters, and the semantic role filler parameters defined in chapter 4. The methods employed to estimate these parameters are not without their flaws, but they are reasonable estimation methods. Table 7.2 provides a summary of what parameters are used in the model and how they are estimated.

Locality Parameters

For each constituent β in a grammar, the grammar must specify the the likelihood that β is expressed versus omitted, and assuming β is expressed, the likelihood of β

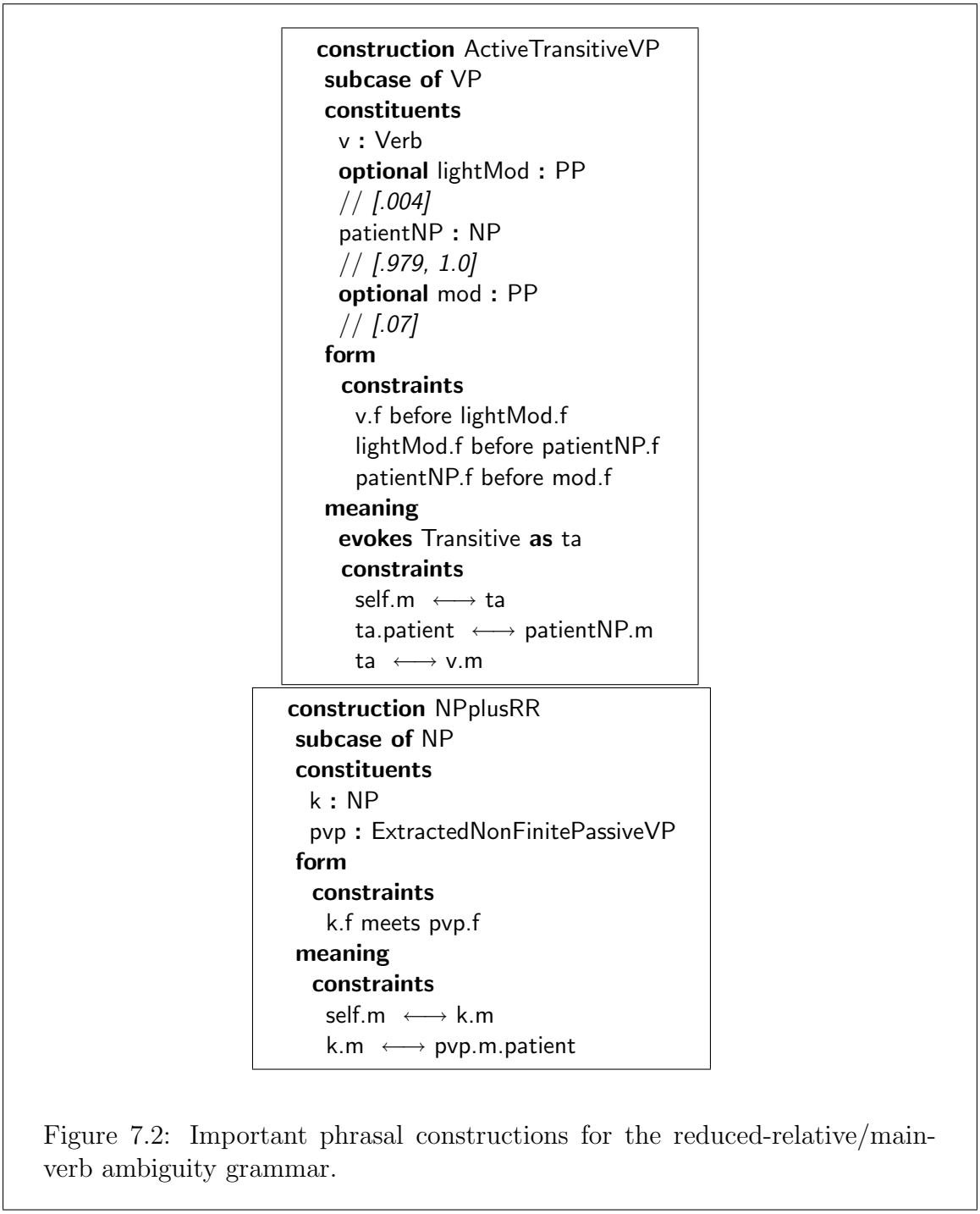

Figure 7.2: Important phrasal constructions for the reduced-relative/main-verb ambiguity grammar.

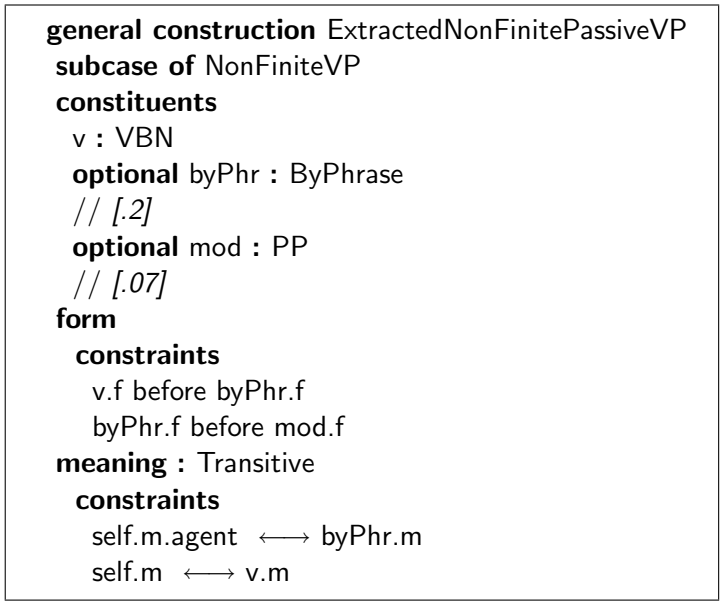

Figure 7.3: The ExtractedNonFinitePassiveVP construction

Parameter	Source	Example
Locality	Treebank	$\Pr(\, modifier_{exp} \mid TransVP \,)$
Constituency	Treebank + Propbank	$\Pr(\, NPplusRR \mid NP \,)$
Semantics	McRae et al.	$\Pr(\, agent \mid cop, arrest \,)$

Table 7.2: A table describing each of the kinds of parameters used in the analyzer and the sources of information used to approximate them. For example, the constituency probabilities are estimated using a PCFG assumption ($\Pr(\, filler \mid type \,)$) and counting up the ratio of the rules similar to the filler over the rules similar to the type constraint.

being locally expressed must also be specified. While most of the constituents in the grammar are required to be expressed locally 100% of the time, figure 7.2 shows that construction ActiveTransitiveVP has a constituent patientNP that can be omitted about 2% of the time.

To estimate the parameters for ActiveTransitiveVP, I turned to the Penn Treebank [49] and Propbank [61]. To estimate the likelihood of the patientNP being omitted, I used PropBank. Summing over of the verbs in the McRae et al. data, I counted the number of times ARG1 was not expressed and normalized by the total number of times that the verbs were used. To estimate the probability of the modifiers, I looked at a table of treebank productions, using the following formulas to estimate the probability of their expression:

$$\Pr(\, exp_{lightMod} \mid ActiveTransitiveVP\,) = \frac{\mid VP \rightarrow V\ PP\ NP... \mid}{\mid VP \rightarrow V...NP... \mid} \quad (7.1)$$

$$\Pr(\, exp_{mod} \mid ActiveTransitiveVP\,) = \frac{\mid VP \rightarrow V\ NP...PP... \mid}{\mid VP \rightarrow V...NP... \mid} \quad (7.2)$$

Equation 7.2 calculates the probability of an optional modifier between the verb and the NP to be the normalized sum of the rule counts in which the rule had a PP in between the V[2] and NP. Similarly for the post-NP modifier, I used the normalized number of times a PP followed an NP. Additionally, the rules were constrained to have a single NP on the right hand side.[3]

To estimate the likelihood of the ByPhrase in construction ExtractedNonFinitePassiveVP, I again looked at the propbank annotations. Propbank annotates verbal targets for voice as well as its semantic arguments. Thus to estimate the likelihood of a by-phrase, I used the normalized count of the verbs in passive voice that have ARG0 specified. My reasoning being that if the verb is passive and the agent is specified, that it is most likely through a by-phrase.

[2]I use V here as a shorthand for all the finite verbal part of speech tags VBD,VB and VBZ.

[3]Distinguishing between NP objects and NP temporal modifiers is difficult given the unannotated treebank trees, so I chose a simple (incorrect) heuristic for estimating the likelihood. For example, *[went]V [to the store]PP [yesterday]NP* is counted as an instance of a light modifier even though it obviously is not.

Constituent Filler Parameters

For the constituent filler probabilities, I use a context free simplifying assumption. Instead of conditioning on the construction and constituent, I just used Pr($filler = \lambda \mid$ *constituent type constraint*). To get PCFG estimates for the constituent filler probabilities, I again used the treebank. For the parts of speech, I just used Pr(*word* | *tag*). For noun phrases, the simple estimate of (for example) the number of pronouns over the number of NPs was used for the likelihood of a pronoun given an NP type constraint.

More care has to be taken for the phrasal constituent filler probabilities because those parameters are important for determining the syntactic priors for the two competing interpretations. For example, the NPplusRR constructions expects an ExtractedNonFinitePassiveVP, and the ExtractedNonFinitePassiveVP construction has at least the transitive passive and the ditransitive passive constructions as concrete subtypes. To count the likelihood of the transitive passive given the passive type constraint, I sum up the number of $VP \rightarrow VBN...$ rules that do not have an NP, SBAR or S following the VBN, and normalized by the total count of $VP \rightarrow VBN...$ rules.[4] Of course the same caveat about NP temporal modifiers applies here as well.

To calculate the likelihood of the NPplusRR construction given NP, I summed the counts of all $NP \rightarrow NP...VP...$ rules and normalized. Similarly for the determiner-noun construction, the normalized sum of the rule counts in which the rule was $NP \rightarrow DT...N$ was used as the likelihood of the determiner-noun construction given NP.

To estimate the likelihood of the FiniteActiveTransitiveVP given FiniteVP, I again used the propbank annotations. Propbank tracks whether a verb is a finite use or not. The sum of all finite annotated verbs was the normalizer. To count the number of finite transitives, I required the verb to be finite, and both ARG0 and ARG1 be present.

[4]That don't have SBAR or S in them

Semantic Parameters

Consider again the pair of example *arrest* sentences taken from McRae et al.:

1. The cop arrested by the detective was guilty.

2. The crook arrested by the detective was guilty.

It should be obvious that the semantic fit of *cop* with the agent and patient roles of arrest should be extremely important to modeling the data. The same is true for the semantic fit of *crook* and the agent/patient roles of arrest. To model the semantic fit, McRae et al. performed norming studies to get human judgments. With a bit of numerical massaging[5], the McRae et al. norming studies can be interpreted as providing:

$$\Pr(\, \textit{likelihood of cop being the agent of arrest} \mid \textit{cop}, \textit{arrest}\,)$$

Thus the semantic parameters for the main effect at *arrested* are provided by McRae et al.

There are two other semantic parameters in these sentences that affect the likelihood of each interpretation. If the reader treats *arrested* as the main verb, *by the detective* should be interpreted as a locative phrase. If the reader interprets *arrested* as the beginning of a reduced relative, then *by the detective* is interpreted as specifying the agent of the arrest frame. To better model the reading time difference at word *was*, I estimated parameters for the semantic fit of *detective* to the landmark role of a TrajectorLandmark schema and a parameter for the semantic fit between *detective* and the agent role of arrest.

Unfortunately, McRae et al. do not provide enough information to estimate these parameters. They do mention that the participants in their norming study averaged 4.6 when asked about the agenthood of the object of the by-phrase (e.g. detective). The objects of the by-phrase are worse agents (on average) than the initial good-agent

[5]The norming studies provide a number α between 1 and 7 that estimates the fit of agent to cop and arrest and a second number β between 1 and 7 to estimate the fit of patient to cop and arrest. Using their α and β, I estimate the likelihood of $\Pr(\, \textit{agent} \mid \textit{Arrest},\ \textit{cop}\,)$ to be $\frac{\alpha-1}{\alpha-1+\beta-1}$.

NPs (e.g. cop), suggesting that the average probability of the good agent NP (e.g. cop) being assigned agent (.83) should be the upper bound for the average probability of the object of the by-phrase (e.g. detective) being assigned agent. Without any further information, I arbitrarily chose .75 for the semantic fit of the object of the by-phrase to the agent role.

For the likelihood of *detective* filling the landmark role of TrajectorLandmark, there is even less information to go on. McRae et al. use a norming value of 1 for their model saying that the objects of the by-phrases *e.g. detective* are poor locations. However, it seems that they are too quick to dismiss the person-for-location metonymy which allows:

1. He ran past the detective
2. He walked by the detective without stopping
3. He threw his knife at the detective
4. Handcuffed, the crook stood defiantly between the two detectives

Even though *detective* can productively be used as a landmark, intuition suggests that there is a bias for animate entities being bound to the trajector role. Without further guidance, I chose .3 as the value of Pr($landmark \mid detective, TrajectorLandmark$).

7.4 Linking the Analyzer's Probabilistic State to Reading Time

The state of a probabilistic parser is at best an indirect indicator of reading time difficulty. As a consequence, various heuristics have been developed to link the state of a probabilistic parser with reading time predictions. As was already mentioned, Pado links processing difficulty to disparity in disagreement between the semantic and syntactic models.

Another heuristic first suggested by Hale [29], is known as surprisal. Surprisal is a heuristic that assumes a correlation between reading time difficulty and the

conditional probability of the incoming word w_i. If w_i has a low probability, then it is assumed that reading time increases. If the probability of w_i is high, then very little reading time should be necessary. Both Levy [46] and Narayanan and Jurafsky [57] have used surprisal to predict difference in reading time, with Narayanan and Jurafsky using it to model a fragment of the McRae et al. data.

The heuristic I employ here is a simplification of a second heuristic developed by Hale [30]. The key idea behind the approach is to calculate the *entropy* over derivations at each word. Entropy is a measure of the uncertainty in a probability distribution. Given a random variable X, entropy is defined as:

$$H(X) = \sum_{x \in X} \Pr(x) \times log \frac{1}{\Pr(x)} \tag{7.3}$$

Equation 7.3 sums over each possible outcome x in X, scaling the log of $1/\Pr(x)$ by the probability of x. Large values of $H(X)$ mean high uncertainty, and small values of $H(X)$ mean less uncertainty.

As the analyzer processes each word of the utterance, it maintains a distribution over interpretations. This distribution estimates the conditional likelihood of each interpretation given the sentence so far. Using equation 7.3, the system calculates the conditional entropy over interpretations at each word. The intuition is that a low entropy value means that the analyzer is fairly certain about which interpretation is the right one, and a high entropy meaning that many viable candidates still remain.

Like Hale, I claim that the change in entropy between words correlates with processing difficulty. Using $H(A)_i$ to mean the conditional entropy over analyses at word w_i, I define the change in entropy at word w_i[6] as:

$$\Delta_i = AbsoluteValue(H(A)_i - H(A)_{i-1}) \tag{7.4}$$

In equation 7.4, the magnitude of the change in entropy (Δ_i) is calculated as the absolute value of the difference in conditional entropy[7] between word w_i and word w_{i-1}.

[6] My definition of the change in entropy is a simplification of Hale's definition.

[7] I define $\Delta_0 = H(A)_0$.

Difference in entropy corresponds to more processing time than maintaining the probabilistic status quo under the assumption that shifting attention is expensive. In short, I claim that both going from unsure to sure incurs processing difficulty and going from sure to unsure also incurs processing difficulty. Although I have no biological evidence for such a claim, this hypothesis is consistent with what Narayanan and Jurafsky [55] refer to as *Attention Principle*.

7.5 Cop and Crook Traces

The differences between the cop trace and the crook trace show qualitatively how the processing differs between the two cases. This qualitative difference motivates the the choice of Δ_i as the means of linking the state of the analyzer with processing difficulty.

For brevity, the traces use abbreviated construction names. "ENFPVP", for example, is short for ExtractedNonFinitePassiveVP. Constituent lightMod is also abbreviated to "lMod", construction ActiveTransitiveFiniteVP is abbreviated to "TransVP", and schema TrajectorLandmark is shortened to "TL".

Consider the analyzer trace for *the cop arrested by the detective was...* shown in figure 7.4. The stacks for words *the* and *cop* are not shown because only a single analysis of probability one is maintained. When the word *arrested* is encountered, two stacks must be maintained, one for *arrested* as a main verb and the other for *arrested* as the beginning of a reduced relative. Stack1 binds Cop to the agent role, Stack2 binds Cop to the patient role. Importantly, the main verb interpretation is dominating both on syntactic and semantic grounds which leads to very little change in entropy at *arrested*.

When the word *by* arrives, the interpretation ranking is reversed. The reduced relative becomes four times more likely than the main verb interpretation because *by* arriving immediately after the ambiguous verb is a reliable indicator that the verb is a past participle. The main verb interpretation has to assume that *by* is the beginning of a locative modifier. The radical shift in the distribution over interpretations results in a large difference in entropy from the last word.

For the most part, when *the* arrives, the status quo is maintained with one exception. A third stack is generated which corresponds to the interpretation of *by* as the beginning of a locative phrase (instead of a byPhrase) within the reduced relative construction. The difference in entropy is minimal.

When the analyzer processes *detective*, the reduced relative interpretation gains further momentum since *detective* is a better agent of arrest than landmark. The likelihood of the reduced relative interpretation increases from .82 to .92. The increased dominance of the reduced relative interpretation decreases the entropy.

Finally, the word *was* makes it syntactically impossible for *arrested* to be treated as a main verb. Stack1 ceases to exist, and the reduced relative interpretation becomes a near certainty. Because the main verb interpretation is removed from consideration, the entropy is again decreased by a substantial amount.

The analyzer trace for *the crook arrested by the detective was...* is shown in figure 7.5. Structurally, the stacks in figure 7.5 evolve the exact same way as they evolve in figure 7.4. The probability distribution over stacks, however, is extremely different.

At *arrested*, there is a huge increase in entropy because of the relative uncertainty between the main verb interpretation and the reduced relative interpretation. As Pado [59] observes, the syntax and semantics disagree about the appropriate interpretation of the sentence.

But by the time *by* is encountered, there is no longer a disagreement between the syntax and semantics. Word *by* makes the reduced relative interpretation both a syntactic and semantic certainty. The probability of the reduced relative interpretation increases to one, and all the entropy at *arrested* is removed, making the decrease in entropy extremely large.

After *by*, the status quo is maintained with the exception of the introduction of the reduced relative with a locative modifier (Stack3) at *the*. The primary reduced relative interpretation maintains a probability of one. Thus when *was* is encountered, no noteworthy change in entropy takes place.

These traces show qualitatively how change in entropy after each word correlates with changes in the belief state over analyses. In the next section, I will show how averaging the change in entropy across the good agent and good patient cases predicts

Word 3: "arrested" . . . Change in entropy:.063
Stack1: Root[S], NPVP[FiniteVP], Arrested-VBD[]
Sem: Arrest.agent = Cop; Probability = .99
Stack2: Root[S], NPplusRR[ENFPVP], Arrested-VBN[]
Sem: Arrest.patient = Cop; Probability = .01

Word 4: "by" . . . Change in entropy:.584
Stack2: Root[S], NPplusRR[ENFPVP], ENFPVP[ByPhr, mod], By[]
Sem: Arrest.patient = Cop; Probability = .83
Stack1: Root[S], NPVP[FiniteVP], TransVP[lMod, patNP, mod], By[]
Sem: Arrest.agent = Cop; Probability = .17

Word 5: "the" . . . Change in entropy: .09
Stack2: Root[S], NPplusRR[ENFPVP], ENFPVP[ByPhr, mod], ByPhrase[NP], The-DT[]
Sem: Arrest.patient = Cop; Probability = .82

Stack1: Root[S], NPVP[FiniteVP], TransVP[lMod, patNP, mod], PP[NP], The-DT[]
Sem: Arrest.agent = Cop; Probability = .17
Stack3: Root[S], NPplusRR[ENFPVP], ENFPVP[ByPhr, mod], PP[NP], The-DT[]
Sem: Arrest.patient = Cop; Probability = .01

Word 6: "detective" . . . Change in entropy: -.31
Stack2: Root[S], NPplusRR[ENFPVP], ENFPVP[ByPhr, mod], ByPhrase[NP], NP[N], Detective-NN[]
Sem: Arrest.patient = Cop; Arrest.agent = Detective; Probability = .92

Stack1: Root[S], NPVP[FiniteVP], TransVP[lMod, patNP, mod], PP[NP], NP[N], Detective-NN[]
Sem: Arrest.agent = Cop; TL.landmark = Detective; Probability = .07
Stack3: Root[S], NPplusRR[ENFPVP], ENFPVP[ByPhr, mod], PP[NP], NP[N], Detective-NN[]
Sem: patient = Cop; TL.landmark = Detective; Probability = .01

Word 7: "was" . . . Change in entropy: -.383
Stack2: Root[S], NPVP[FiniteVP], Was-VBD[]
Sem: Arrest.patient = Cop; Arrest.agent = Detective; Probability = .995

Stack3: Root[S], NPVP[FiniteVP], Was-VBD[]
Sem: patient = Cop; TL.landmark = Detective; Probability = .005

Figure 7.4: The trace for *The cop arrested by the detective was...*

Word 3: "arrested"... Change in entropy: .944
Stack2: Root[S], NPplusRR[ENFPVP], Arrested-VBN[]
Sem: Arrest.patient = Crook; Probability = .64

Stack1: Root[S], NPVP[FiniteVP], Arrested-VBD[]
Sem: Arrest.agent = Crook; Probability = .36

Word 4: "by"... Change in entropy: -.934
Stack2: Root[S], NPplusRR[ENFPVP], ENFPVP[ByPhr, mod], By[]
Sem: Arrest.patient = Crook; Probability = 1.0
Stack1: Root[S], NPVP[FiniteVP], TransVP[lMod, patNP, mod], By[]
Sem: Arrest.agent = Crook; Probability = .00

Word 5: "the" ... Change in entropy: .106
Stack2: Root[S], NPplusRR[ENFPVP], ENFPVP[ByPhr, mod], ByPhrase[NP], The-DT[]
Sem: Arrest.patient = Crook; Probability = .99

Stack3: Root[S], NPplusRR[ENFPVP], ENFPVP[ByPhr, mod], PP[NP], The-DT[]
Sem: Arrest.patient = Crook; Probability = .01

Stack1: Root[S], NPVP[FiniteVP], TransVP[lMod, patNP, mod], PP[NP], The-DT[]
Sem: Arrest.agent = Crook; Probability = .00

Word 6: "detective" ... Change in entropy: -.061
Stack2: Root[S], NPplusRR[ENFPVP], ENFPVP[ByPhr, mod], ByPhrase[NP], NP[N], Detective-NN[]
Sem: Arrest.patient = Crook; Arrest.agent = Detective; Probability = .99

Stack3: Root[S], NPplusRR[ENFPVP], ENFPVP[ByPhr, mod], PP[NP], NP[N], Detective-NN[]
Sem: patient = Crook; TL.landmark = Detective; Probability = .01
Stack1: Root[S], NPVP[FiniteVP], TransVP[lMod, patNP, mod], PP[NP], NP[N], Detective-NN[]
Sem: Arrest.agent = Crook; TL.landmark = Detective; Probability = .00

Word 7: "was" ... Change in entropy: -.01
Stack2: Root[S], NPVP[FiniteVP], Was-VBD[]
Sem: Arrest.patient = Crook; Arrest.agent = Detective; Probability = 1.0

Stack3: Root[S], NPVP[FiniteVP], Was-VBD[]
Sem: patient = Crook; TL.landmark = Detective; Probability = .00

Figure 7.5: The analyzer trace for the *The crook arrested by the detective was...*

sentence position	Good Agent	Good Patient
verb	.14 (21)	.79 (51)
by	.28 (42)	.75 (48)
was	.24 (36)	.02 (01)

Table 7.3: The magnitude of the average change in entropy (average Δ_i) at the verb, at "by", and at "was" for the two test conditions. The number in parentheses is the percentage of the total change in entropy summed across the three sentence positions. For example, the change in entropy at the verb for the good agent case is .14, and .14/(.14+.28+.24) = .21.

reading time difficulty consistent with the trends in the McRae et al. data.

7.6 Results

The system analyzes each of the sentences tested by McRae et al. and after each word in the sentence, the entropy and change of entropy are calculated using the equations 7.3 and 7.4. For each word w_i, Δ_i is averaged separately for all the good agent cases and for all the good patient cases. This provides the magnitude of the average change in entropy at the verb, at "by", and at "was" for both test conditions which are shown in table 7.3.

McRae et al. present the sentence two words at a time, first showing the initial NP (*the cop*), then the past participle + by (*arrested by*), then the agent NP (*the detective*), and finally the actual main verb (*was guilty*). As is shown in table 7.1, they provide the reading time differences over the unreduced baseline after the past participle + by, after the agent NP and after the main verb. Following Narayanan and Jurafsky [55] and Pado [59], the analyzer state after the main verb is matched to McRae et al.'s past participle + by effect, the analyzer state after *by* is matched up with McRae et al.'s agent NP effect, and the analyzer state after *was* is matched up with McRae et al.'s reading time effect.

Instead of comparing the Δ_i directly with the reading time difficulties found by

McRae et al., I compare the percentage of difficulty. For the analyzer, the percentage of difficulty is the Δ_i, normalized by the sum of the Δ_i's at the three points of interest. As shown in table 7.3, for good agents at the verb, the average Δ_{verb} value is .14. To get the average percentage of difficulty, I normalize .14 by .14+.28+.36 and then multiply by 100. A similar computation is done for the McRae et al. reading time difficulties shown in table 7.1 to get the observed average percentage of difficulty.

Figure 7.6 compares the average percentage of difficulty predicted by the analyzer (the solid line) with the observed average percentage of difficulty taken from McRae et al. (the dashed line) for the good agent cases. The analyzer appropriately predicts about 20% of the reading time difficulty at the past participle. The analyzer predicts a slightly higher percentage of reading time difficulty at *by* (42 vs 38), and it predicts a slightly lower percentage of difficulty at *was* (36 vs 40). For the good agent cases, the analyzer predicts the highest amount of difficulty at *by* because *by* is a strong cue for the reduced relative thereby leading to a large change in entropy. At *was*, the main verb interpretation is finally rendered impossible, in spite of the strong semantic bias, and therefore the difference in entropy is large.

Figure 7.7 compares the average percentage of difficulty predicted by the analyzer (the solid line) with the observed average percentage of difficulty taken from McRae et al. (the dashed line) for the good patient cases. The analyzer correctly predicts that the greatest amount of difficulty is at the past participle (51 vs 63), that the difficulty at *by* is lower (48 vs 36), and that difficulty as *was* is nonexistent.

While the trends match up, the analyzer has trouble predicting the smaller percentage of difficulty at *by*. At the past participle, there is a lot of entropy in the analyzer's distribution over interpretations because the syntax and the semantics disagree. The syntax prefers the main verb interpretation while the semantics prefers the reduced relative interpretation. Then at *by*, almost all the uncertainty goes away because the syntax and semantics both overwhelmingly prefer the reduced relative interpretation. Thus another large (albeit somewhat smaller) change in entropy takes place. At *detective*, the reduced relative interpretation is a near certainty, and there is no change in entropy when *was* is encountered. This predicted lack of difficulty is consistent with the lack of difference over the baseline found by McRae et al.

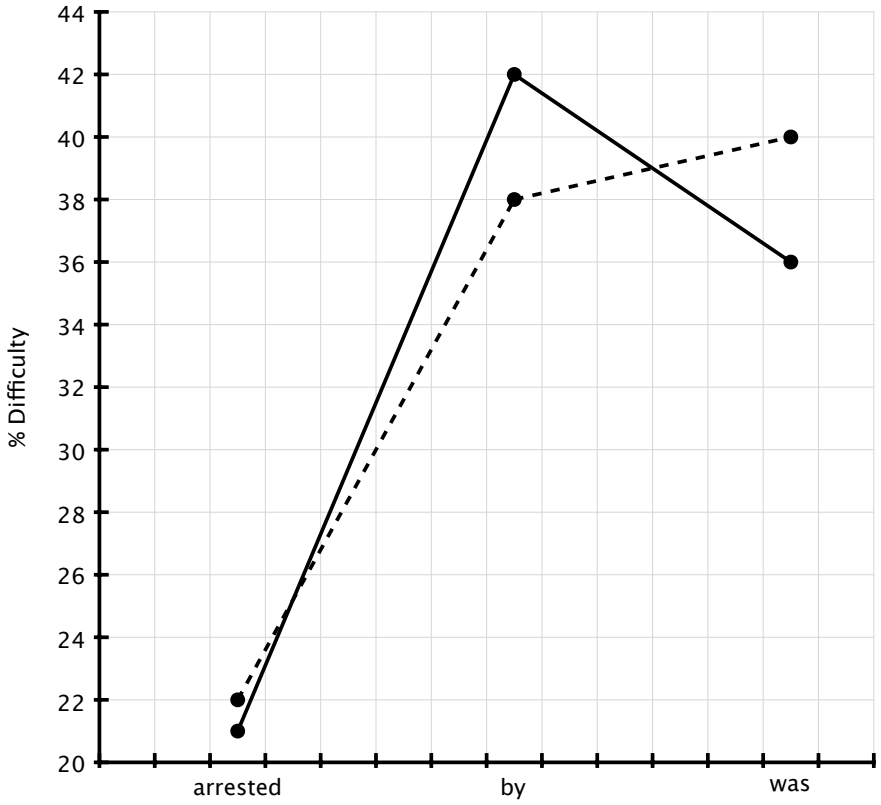

Figure 7.6: For good agent cases: The percentage of reading time difficulty predicted by the analyzer (the solid line) and the observed percentage of reading time difficulty (the dashed line) for the good patient cases.

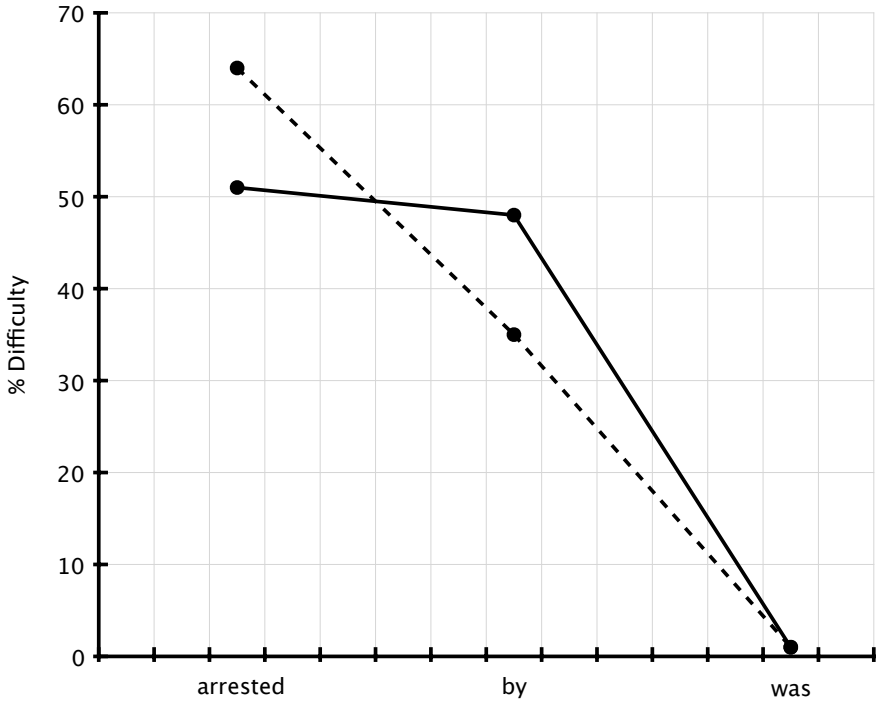

Figure 7.7: For good patient cases: The percentage of reading time difficulty predicted by the analyzer (the solid line) and the observed percentage of reading time difficulty (the dashed line).

McRae et al. also describe a sentence completion task. The sentence prefixes were to be continued by by the subjects of the experiment, and the experimental stimuli were similar to the ones used in the reading time study. For example, a subject in the sentence completion study would be given the prefix *the cop arrested* and have to write the rest of sentence. For both good agents and good patients, McRae et al. provided prefixes like:[8]

- the cop arrested
- the cop arrested by
- the cop arrested by the
- the cop arrested by the detective

McRae et al. then measured the proportion of reduced relative completions at the past participle, at *by*, at *the*, and at the agent noun. The results of their study are shown in figure 7.8. For the subjects in their study, the proportion of reduced relative completions is always higher for good patient cases, and for both good agents and good patients, the proportion of reduced relatives goes up dramatically at *by*.

Using the conditional probability of the reduced relative interpretation after each word, the analyzer can also make predictions about the proportion of reduced relative completions. These predictions are also shown in figure 7.8. The analyzer's predictions follow the trends found by the McRae et al. sentence completion study. Comparing the analyzer's predictions to the human predictions provides some insight on the accuracy of the analyzer's parameters. From the fact that the probability of the reduced relative is higher for good patients at the past participle than the proportion of human completions (.36 vs .2) suggests that the parameters used for analysis make the prior probability of the reduced relative too high. For good agents, the human proportion of reduced relative completions at *by* is much lower than the analyzer's probability of a reduced relative at *by*. This suggests that *by* is not as reliable of a cue for the reduced relative as the analyzer's parameters suggest.

[8]No subject saw the same initial NP + past participle more than once.

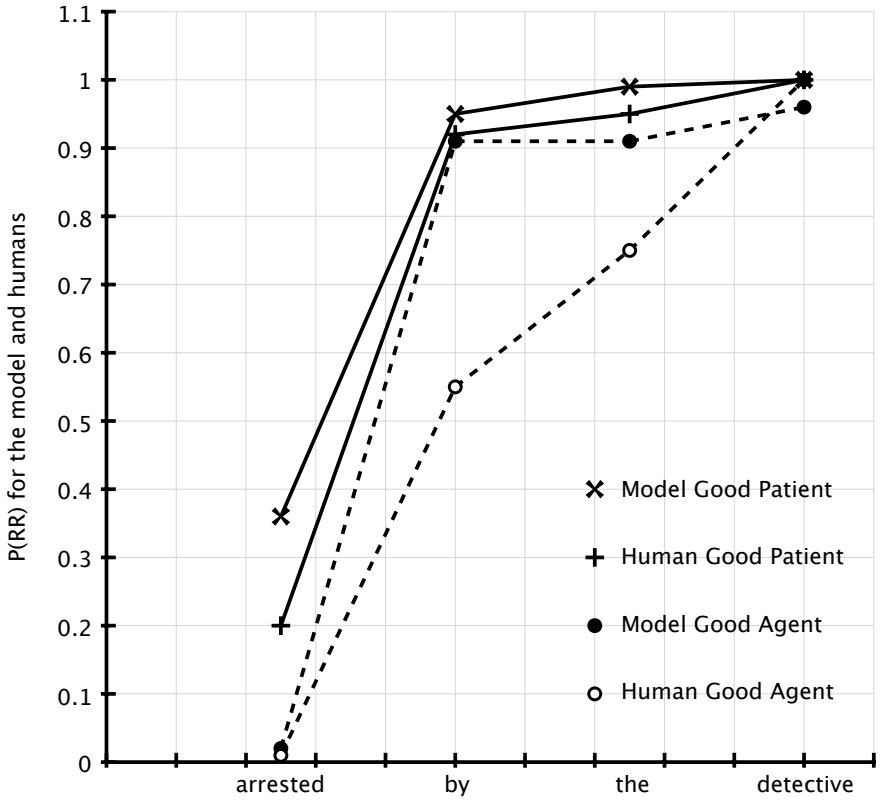

Figure 7.8: For humans, this figure shows the proportion of reduced relative sentence completions for good agents and good patients found by McRae et al. For the analyzer, it shows the conditional probability of the reduced relative given the prefix.

7.7 More Modeling

The reading time predictions provided by the analyzer are the best McRae et al. predictions made by a probabilistic model to date. Of course they are also the only predictions made by model of deep interpretation. This suggests that a construction-based model of deep interpretation can act both as a cognitive model of sentence processing and as a psycholinguistic model of human sentence processing. Though more experiments must be modeled before the constructional analyzer can be truly be considered a unified model of sentence processing.

Chapter 8

Analyzing Child-Directed Utterances in Mandarin

This case study describes the application of the constructional analyzer to the interpretation of child-directed utterances in Mandarin. Mandarin data provides an interesting test for the analyzer because Mandarin speakers seem to rely more on context to communicate meaning than English speakers. For example, constituent omission is a productive phenomena in Chinese. Thus the extensions to the analyzer for productive omission prove their utility in this chapter because they enable the system to parse the child-directed utterances with omission.

Additionally, the scale of this task is much greater than for the other two case studies in that the number of utterances the system analyzes is an order of magnitude larger than in other tests. Efficiency is also important because the Mandarin grammar is much larger and more ambiguous than any other grammar used with the system. This leads to a much larger search space, and thus the best-fit metric becomes extremely important for focusing the analyzer on promising analyses.

This chapter is structured as follows. Section 8.1 provides a summary of the differences between English and Mandarin, including a discussion of productive omission. Section 8.2 describes the analyzer's role in a model of child language learning. Section 8.3 covers the extensions to the analyzer to allow for multi-rooted analyses. Section 8.4 describes the training and validation as well as the grammar used to model

that data. Section 8.5 shows how examples from the Mandarin training corpus are analyzed and related to context. Section 8.6 shows a proof-of-concept method for estimating the constructional parameters, which improve the speed and accuracy of the analysis process.

8.1 Mandarin and Productive Omission

Mandarin and English are similar in that they are both word-order dependent languages that do not employ a rich morphology. From my English speaker's perspective, though, there is a striking difference between Mandarin and English: Mandarin speakers do not have to say as much. In Mandarin, one is allowed to freely omit constituents whenever the meaning of the constituent can be inferred from the situational or discourse context. This means that for Mandarin speakers, constituent omission is a productive phenomena, and it is used much in the same way as English speakers use pronouns.

Even in the child-parent interactions that make up the Beijing CHILDES corpus [83], productive omission is extremely common. In a pilot study of the the central ditransitive construction was used in the Childes transcripts [53], Mok and Bryant note that all eight[1] of the possible omission patterns are found in the data. Five of these patterns are shown in figure 8.1 transcribed in standard PinYin. Note that because Mandarin is a tonal language, each of the words in the transcribed utterances are followed by numbers that indicate the tone of the word.

Mok and Bryant calculate the frequency of each of the omission patterns, as well as the frequency of each argument being omitted. They find that:

- The *giver* is omitted approximately 80% of the time.
- The *theme* is omitted approximately 60% of the time.
- The *recipient* is omitted approximately 40% of the time.

[1]The *giver* can be expressed or not expressed, the *theme* can be expressed or not expressed, and the *recipient* can be expressed or not expressed.

ma1+ma gei3 ni3 zhei4+ge .
mother give 2sg this+CLS
Mother (I) give you this (a toy).

ni3 gei3 yi2 .
2PS give auntie
You give auntie [the peach].

gei3 a1+yi2 yi2+ge qu4 ao .
give auntie 1+CLS DEITaway EMP
[You] Give auntie one (peach).

ao ni3 gei3 ya .
EMP 2PS give EMP
Oh you give [auntie] [the peach].

gei3 .
give
[I] give [you] [some peach].

Figure 8.1: Examples from the Beijing CHILDES corpus illustrating productive omission with the Mandarin ditransitive construction. For these examples, the Chinese characters have been transcribed as pinyin. A literal translation of each word is provided as well as a paraphrase in italics. Bracketed phrases in the paraphrase are omissions in the original utterance. Parenthesized phrases are the referents of pronouns.

- All of the arguments are omitted about 30% of the time.
- All of the arguments are expressed about 6% of the time.

Thus Mandarin language learners are not often given the full construction, and yet still learn that they can use all three arguments when necessary.

8.2 Language Learning and "Robust" Parsing

In a grammar learning framework, the parser and the language learning module (learner) form a loop. For each input utterance, the parser does the best it can with

the current (partial) grammar, and the language learning module uses the output of the parser to improve the partial grammar. Chang [8] instantiates this framework for English argument structure construction learning in ECG. In her model, each input utterance is paired with a description of that utterance's meaning. The input/situation pairs are intended to be analogous to a child hearing the utterance in a given situation. For each (utterance,situation) pair (α, β), Chang uses a simple construction parser built by Bryant [5] to parse utterance α. Her model then compares the situation β to the (partial) understanding of the utterance γ provided by the construction parser. The learning module hypothesizes new constructions to cover the difference between γ and β. Proposed constructions that are frequently used become permanent additions to the grammar, while proposed constructions that are not frequently used are removed.

Mok [52] extends Chang's model to be more psychologically and cognitively plausible, and to learn Mandarin constructions with argument omission as well. Mok uses the constructional analyzer described in this thesis as the parsing module in her work. She also incorporates a third component into the learning loop. This third component is a context module that tracks the ongoing situational and discourse context after each utterance and fits the returned analysis to the current context. The context module enables the learning of constructions with argument omission.

8.3 Extensions for Grammar Learning

The grammar learning scenario puts extra constraints on the design of the system. The in-progress grammar will be incomplete, and if the grammar is incomplete, then often there will not be a single construction that spans the whole analysis. To deal with this problem, I extended the system to return partial analyses in which a set of fully-matched constructions can be used to cover the input. I refer to a set of analyses that each cover a subspan of the input as a *multi-rooted* analysis.

Multi-rooted analyses have the following constraints:

- All returned roots must be complete (all constituents must be marked as omit-

ted, matched or nonlocal)

- Every subspan of the utterance must be covered.
- A word cannot be a leaf of more than one root.
- To allow skipping, a root α can have a span that is within the span of another root ρ. In this case, root ρ will not have any shared words with α, but will consider it to be something like an interjection and skip it. For example, with the input *the the man man*, the analyzer can return a multi-rooted analysis with a span of (0,4) and the other with (1,3), and both NPs are treated as valid sub-analyses of the utterance.

Using a scheme similar to that of the robust left corner parser described by Rose and Lavie [69][2], the strategy I employ allows for multiple roots and skipping according to the constraints specified above. To implement the strategy, two additional parser operations for switching between roots have been added to the system. Importantly, these operations have been defined such that a set of complete roots has exactly one derivation in the analyzer.

Using the stack notation from chapter 5, the operations are defined as:

- **New Root**: Makes a new stack to incorporate the next word θ.

$$ROOT_0[\beta_0\gamma_0], \ldots, \alpha_j[] \quad \textbf{becomes} \quad ROOT'_0[\beta'_0\gamma'_0]\ LS = \alpha_j, \theta[]$$

The current stack is rooted by $ROOT_0$ and has α_j as its top. Item α_j is required to be a complete lexical item and additionally, α_j cannot already have been assigned a stack as its right sibling. If all those conditions are met, then a new root $ROOT'_0$ is built. The notation $LS = \alpha_j$ indicates that the left sibling of the stack rooted by $ROOT'_0$ is the stack that has α_j as its top item. Note that $ROOT_0$ could also have a left sibling.

[2]Their LCFlex parser has also been used to parse utterances from the Childes corpus [72].

Whenever a new root is created, a user-set penalty term is added into the cost of the derivation. Additionally, the conditional likelihood of pushing θ given $ROOT'_0$ is also incorporated into the derivational cost. The semantic cost of the new root is the total semantic cost of α_j and its left siblings' semantic cost plus the semantic cost of the new stack.

- **Switch Back**: If the current stack is complete and it has a left sibling, switch to it.

  $ROOT'_0[]\ LS = \alpha_j\ RS = \rho$ **becomes** $ROOT_0[\beta_0\gamma_0], \ldots, \alpha_j[]\ RS = \rho \cup ROOT'_0$

  The current stack is rooted by $ROOT'_0$ and has α_j as its left sibling and a set of right siblings (RS) represented by ρ. The switch back operation switches back to α_j. Item α_j's right sibling set is the union of ρ and $ROOT'_0$.

  There is no cost to perform a switch back operation because it is deterministically required given a complete root. Thus the constructional probability of α_j equals the constructional probability of $ROOT'_0$. In addition to the local[3] semantic cost of α_j, α_j's semantic cost must be updated to include the semantic cost of $ROOT'_0$ and each of the elements of ρ.

The final modification to the parser is to update the heuristic function for computing the probability of the next word w_{i+1}. Assume that the current stack is ψ_j and that ψ_j's left sibling is ψ_{j-1}. I will use $\Pr(\, w_{i+1} \mid \psi \,)$ to refer to the probability of the next word given a stack.[4] Also, assume an additional-root penalty of π and the probability of completing ψ_j to be $comp(\psi_j)$. Then the probability of the next word is:

$$\Pr(\, w_{i+1} \mid \psi_j \,) + \pi \times \Pr(\, push \mid new\ root \,) + \Pr(\, w_{i+1} \mid \psi_{j-1} \,) \times comp(\psi_j) \qquad (8.1)$$

The first term of equation 8.1 is just the normal probability of the next word given the current stack. The second term of the equation computes the likelihood of the

[3]The local semantic cost must also incorporate the semantic cost of $ROOT_0$'s left siblings as well.
[4]This is the same estimate of the probability of the next word defined in chapter 5.

next word assuming a new stack is used to predict the word. The third term is the probability of ψ's left sibling predicting the next word. Before the system can switch back to ψ_{j-1}, it has to complete ψ_j, and thus the probability of the next word given ψ_{j-1} is scaled by the probability of completing ψ_j.

Note that there is a built-in locality assumption in equation 8.1. The equation could have also added in the probability of the next word given ψ_{j-2}, but as written, it obviously does not. Not including ψ_{j-2} makes the assumption that two roots back is too far away to be useful for predicting the next word. This assumption improves the efficiency of the system and also cuts down on the search space, but it does have ramifications for completeness.

To show how the multi-root operations work, assume a simple grammar $S \rightarrow A\ B$, $A \rightarrow a$, and $B \rightarrow b$. Without multi-rooted parsing, the only parseable sentence is *ab*. But ff you allow for multi-rooted parsing as I have described it in this section, sentence *aabb* is also parseable with two roots using the following derivation (where sibling stacks are separated by a ";":

$$S[AB], A[] \rightarrow S[AB], A[]; S[AB], A[] \rightarrow S[AB], A[]; S[B] \rightarrow$$

$$S[AB], A[]; S[B]B[] \rightarrow S[AB], A[]; S[] \rightarrow S[AB], A[] \rightarrow$$

$$S[B] \rightarrow S[B], B[] \rightarrow S[]$$

Sentence *aabbab* is also parseable with three roots. However, sentence *aababb* is not parseable because of the way the probability of the next word function is calculated. The stack arrangement when the final b is encountered looks like:

$$S[AB], A[]; S[B], B; S[B], B[];$$

And the first $S[AB], A[]$ stack is too far back to predict b.

8.4 Data

The Mandarin data used for this case study comes from the Childes Beijing corpus [83]. The corpus is a collection of transcripts taken from taped interactions between

parents and their child in Beijing. The training data for the experiments in this case study consists of 529 utterances from the transcripts of four children all between 20 and 24 months old. The mean length of the 444 non-interjection utterances in the training set is 3.68 words and the longest utterance is 13 words. A validation set was also built, consisting of 173 utterances pulled from the transcripts of the fourth child with the mean length of the 126 non-interjection utterances being 3.23 words. Eva Mok annotated each utterance in the training and validation sets with a a frame-based semantic representation of each utterance as well as the situational and discourse annotation necessary to make sense of the utterance.

In addition to annotating utterances from Beijing Childes, Mok also built an ECG grammar to cover the utterances in the training set.[5] Mok defined 263 schemas for modeling the common scenarios that a two year old encounters (e.g. eating, playing, throwing, being naughty), and defined 174 ontological types in her context model. Paired with those 263 schemas are about 609 constructions of which:

- 106 are abstract and used for defining phrasal and lexical category structure.
- 73 are concrete phrasal/clausal constructions consisting of the argument structure constructions, topicalization constructions, noun-phrase constructions and adjunct constructions.
- 283 are open class lexical constructions and 135 are closed class function words.
- There is also a small set of constructions for left-overs such as the Unknown-Word construction for use when the analyzer has no lexical record for an input form.

The parameters for her grammar are all learned based on the training data as is shown in section 8.6.

8.5 Qualitative Results

To investigate what the system could and could not accomplish with the Mandarin grammar, Eva Mok and I parsed the training corpus without enabling multi-rooted

[5]Lexical constructions for the validation set were also built.

analyses. Because we were interested in the structural aspects of the grammar, and not the parameters, we use uniform constructional parameters and no semantic parameters.

Using the Mok grammar, the analyzer found a parse for 437 of the 530 utterances in the training set and 129 of the 173 utterances in the validation set. Mok performed a detailed analysis of the first 150 utterances in the training set ignoring repeats. Of those 150, 125 of them had a parse given the Mok grammar. Of those 125, 76 of them were perfect matches to the intended interpretation. Eight of the returned analyses chose the right constructions to cover the data, but the context model did not resolve the omitted arguments correctly. For the forty remaining incorrect analyses:

- nine could not be properly analyzed because the necessary constructions were not in the grammar
- 12 used an incorrect word sense or had an attachment problem
- 3 analyses had topicalization issues. 2 required a topicalization that did not happen and the third did not topicalize when it should have.
- 12 had incorrect constituent omissions
- 1 had a problem with reduplication
- 3 had a problem with the word *le* which in sentence final position could either mean perfective aspect or act as the current relative state marker

To make these results more concrete, four of the 125 parsed utterances are shown below. These example analyses show what the Mandarin analyses looked like and how omission works within the system. Two of the four analyses are successful analyses that have omitted constituents. For each omitted constituent (and noun phrase), the context model provided a set of plausible referents that contained the correct referent. The other two analyses shown below are ones for which the analyzer could not find the correct analysis. The bad parses illustrate some shortcomings of the system that can hopefully be corrected by future research.

8.5.1 Successful Analyses

The two good analyses in this section illustrate how the system deals with omission, and how the analyzer uses the context model to resolve the omitted arguments. Additionally, these good analyses illustrate the rich semantics that Mok has incorporated into her grammar. The rich semantics is a highly informative for the context model when it has to choose a referent for a referring expression or an omitted constituent.

Utterance (1) is a simple example in which the mother admonishes the child for mashing a piece of peach into the ground:

(1) hai2 gei3 wo3 cuo1
too BEN 1sg mash
Translation: On top of that, [you're] mashing [it] for me!

This four word utterance omits the subject and the object (shown as bracketed phrases in the gloss). I have glossed the meaning of "hai2" with the colloquial "on top of that", to indicate the exasperation of the mother. The benefactive marker "BEN" is used for both benefaction and malefaction in Mandarin, and when "gei3" is used as the benefactive/malefactive marker[6], it comes before the verb.

Figure 8.2 shows a compressed form of both the constructional tree and the semspec for utterance (1). The proper analysis according to Mok's grammar is an ActiveSubjectVP that combines a subject s with an active vp vc. In this case, the subject is omitted and the VP is a TransVPwithMod The TransVPwithMod construction combines a VP (in this case the CausedMotionVP1) with a special construction that gathers all the preverbal modifications together called PreverbalModifications. Construction PreverbalModifications has constituents for an adverbial phrase like the construction that covers "hai2" as well as the construction PreverbalGei3Phr which covers the "for me" part of the utterance.

Omission is marked in the constructional tree as skolem constructional types that are in the analysis, but left unfilled. For example, constituent s with index 20 has a skolem referring expression with nothing but an m role because the subject of the

[6] "gei3" is also the word for "give" in Mandarin

utterance has been omitted. Similarly, the CauseMotionVP1's n constituent (indexed by 50) is omitted.

For both of the omitted constructions, the context model returns a candidate set of referents. The system assumes that the referents of omitted constructions must be contextually very salient, and this limits the scope of possible referents. For the omitted subject of type Animate, the model returns the two salient animate entities in context: the child and the mother. The omitted object has type ConcreteEntity, and the range of possible referents (just given the type constraint) is much broader. Thus the system returns a larger set of referents including the child, the mother, the peach, the peach's peel, the ground, and the room that they are in. In a more sophisticated context model, more aspects of the semantics of the utterance would play a role in determining a more accurate set of referents.[7] Additionally, the pronoun "wo3" (me) specifies that it resolves to the speaker, so the context model returns the mother.

The Mash schema is treated as a force application with roles for force supplier (forcesupp) and the force recipient forcerecip. In utterance (1), the Mash schema fits into a larger CauseMotion scene, with the Mash itself being the cause and the effect being a TranslationalMotion in which the force recipient is the mover. The meaning of the benefactive phrase is represented simply as a BeneMalefaction schema with the "benefit" being the CauseMotion scene and the "beneficiary" being the meaning of the PreverbalGei3Phr''s np which in this case was resolved to the mother.

Utterance (2) is another simple example in which the mother admonishes the child for putting too much lotion on her forehead:

(2) bie2 mo3 wai4+tou2 a
NEG apply forehead SFP
Translation: Don't apply [the lotion] to your forehead

This four word utterance is an imperative that omits the object (shown as bracketed phrases in the gloss). The NEG particle indicates a command to not do something. The sentence final particle "a" reduces the forcefulness of the admonishment.

Figure 8.3 shows the analysis for utterance (2). The top level construction is

[7] Mok has extended the context model with an additional process that fits the utterance's semspec to the current context that also takes the semantics of the utterance into account.

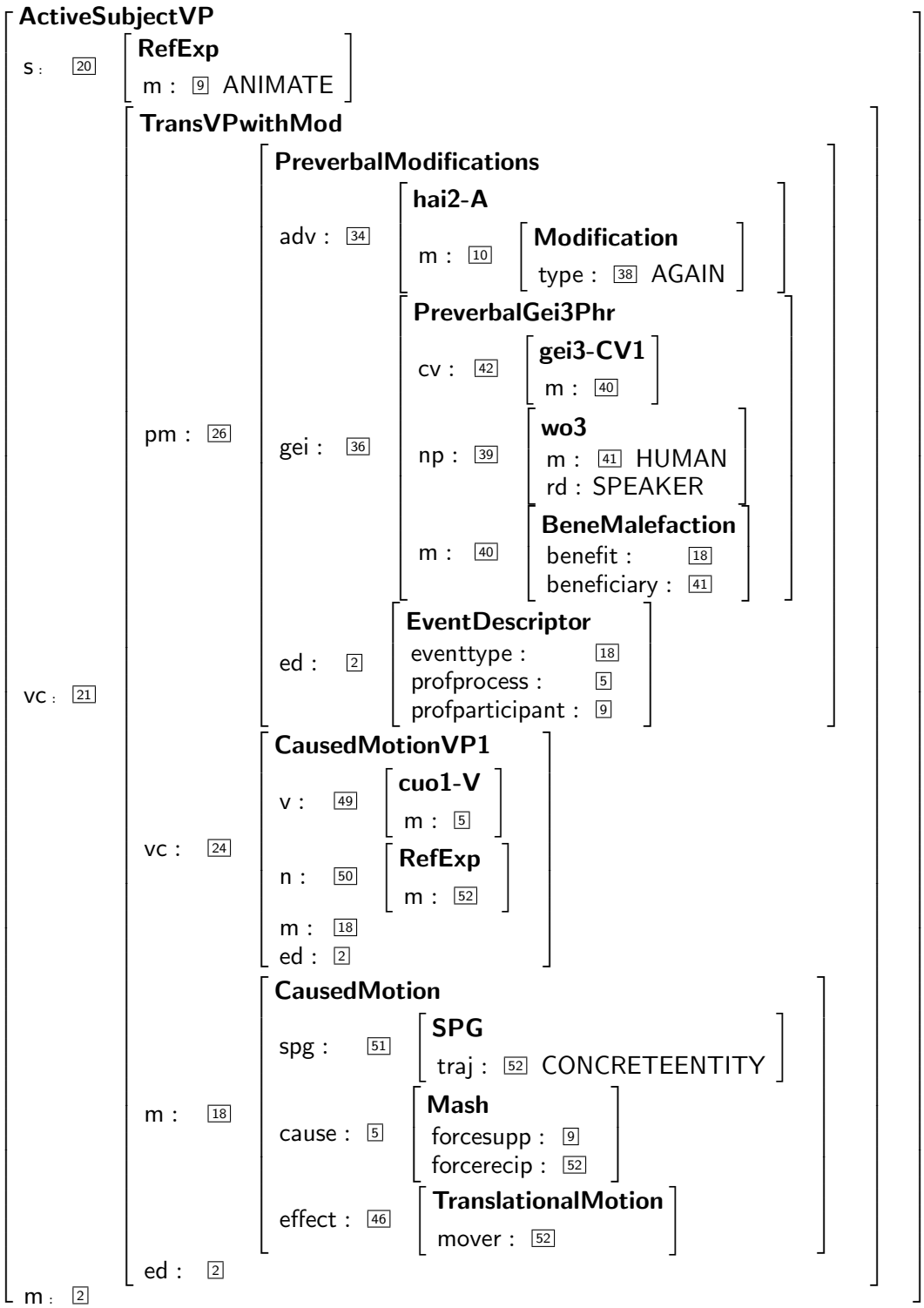

Figure 8.2: The complete analysis of utterance (1).

NegImperativeSFP which is a negative imperative construction that combines the negation marker *bie2* with an ActiveSubjectVP. It then treats the subject constituent (s) as omitted, binding the ADDRESSEE filler into the meaning of the subject.

The meaning of the negative imperative in Mok's grammar is a Negation schema that binds the negated action (in this case an Apply schema instance) to the process slot. Additionally, the bie2 construction evokes a Warning schema, but the Warning schema instance is not shown because of space considerations.

Construction CausedMotionVP3 is special in that it treats its object as the goal location of the CausedMotion scene (the forehead). To do this, a special phrase that turns a common noun into a location (construction CN-Loc) is used. The CausedMotionVP3 construction's constituent n is omitted, and it refers to the theme of the CausedMotion scene which in this case is the substance being applied.

Resolving the two omitted items is easy for the context model because the grammar provides a lot of information in this utterance. The subject referring expression has its meaning bound to ADDRESSEE, and the addressee is the child. The omitted object of the CausedMotionVP3 construction has its meaning bound to SUBSTANCE because of the Apply schema, and the only salient substance in the discourse/situational context is the lotion.

8.5.2 Incorrect Analyses

The analyses in this section show some of the errors that the system makes. Some of the errors are reasonable such as utterance (3) below, and some are a by-product of the fact that the best-fit metric as defined in chapter 4 has a very weak model of semantics and context. For both utterance (3) and utterance (4), incorporating context into the best-fit metric should also make both of these incorrect analyses less likely because the analyzer could prefer analyses that are consistent with the situation. Intonation would also help because it would be easier to distinguish speech acts.

The first incorrect analysis that I will describe is utterance (3) in which the mother tells the child to give a piece of fruit to the Aunt so that the Aunt can eat it:

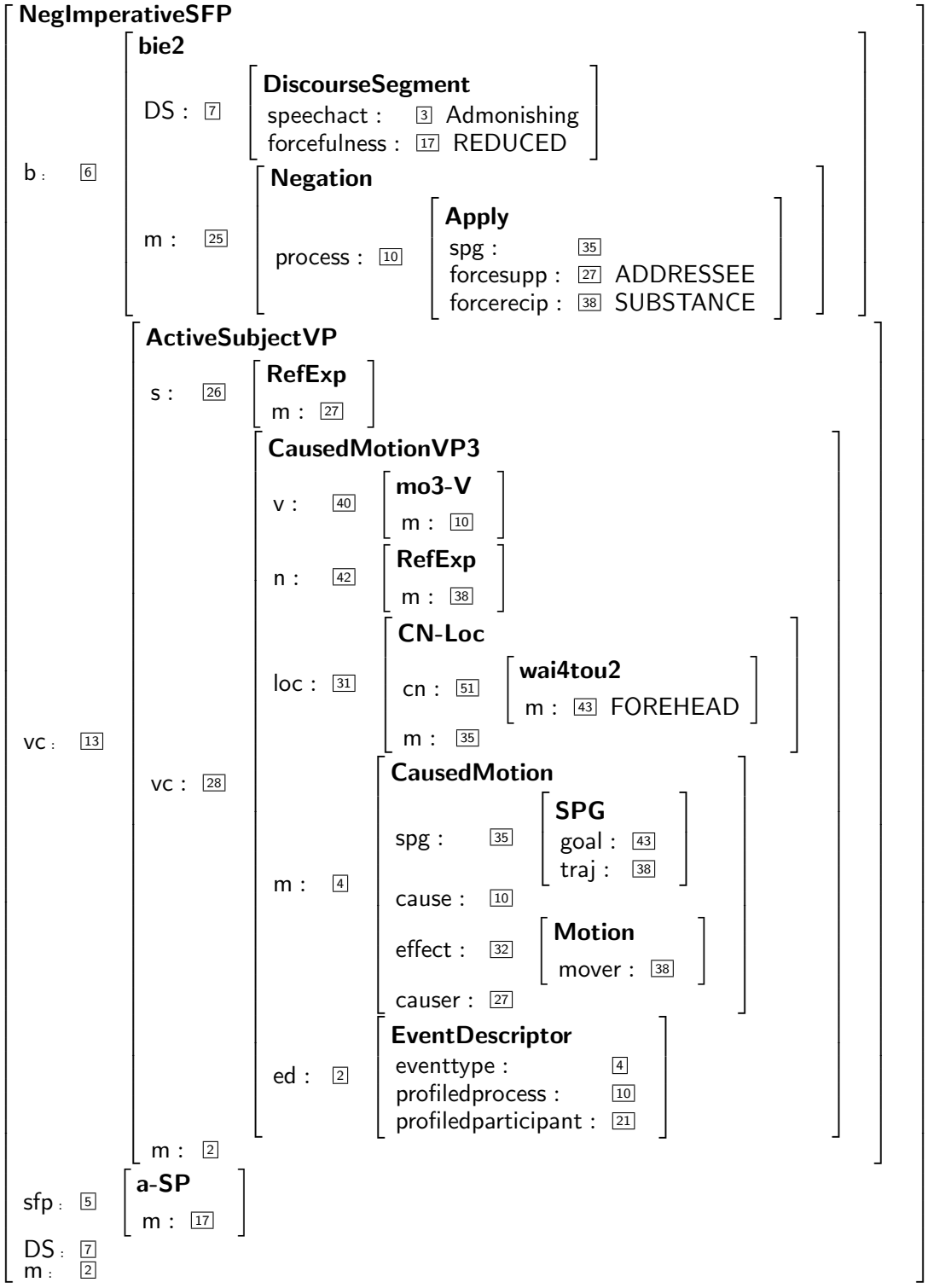

Figure 8.3: The compressed analysis of utterance (2).

(3) gei3 a1+yi2 chi1
give aunt eat
Translation: Give this to Aunt so she can eat it.

For utterance (3), the system should have returned a pivot analysis in which Aunt was both the recipient of "give" and the agent of "eat". Instead it returns a benefactive analysis translated as *eat this for auntie*. Utterance (3) is ambiguous because the verb "give" (gei3) is a homophone with the benefactive marker (also gei3). Utterance (3) looks extremely similar to utterance (1), and the ambiguity making the incorrect returned analysis reasonable both syntactically and semantically. The incorrect analysis is shown in figure 8.4.

Because the analysis in figure 8.4 is quite similar to the analysis in figure 8.2, I will briefly describe it. Like the analysis of utterance (1), the subject and object of the main verb are treated as omitted. Additionally, the primary VP in this example is the IngestionVP which takes verb like *eat*, *chew* and swallow. In this case, the verb is *eat* and the Eat schema is bound into the meaning of the utterance.

For both utterance (1) and utterance (3), both the pivot interpretation and the benefactive interpretation are reasonable on syntactic and semantic grounds. Even with improved constructional and semantic parameters, though, it would be hard for the system to get both analyses right.[8] However, the context of and intonations of each utterance could disambiguate the choice of interpretation. For example, if the child sees the Aunt reaching out to take a piece of fruit, then the pivot construction should be more likely than the benefactive interpretation. Examples such as these illustrate the importance of including more context into the best-fit metric.

Utterance (4) takes place as the mother and child are inspecting a desk lamp. The mother wants confirmation about whether the child sees the desk lamp:

(4) ni3 kan4 dao4 dian4+deng1 le ba
you see ASP ACH light ASP/CRS SFP
Translation: You successfully see the desklamp, right?

For utterance (4), the system should have returned an analysis in which the the particle *dao4* denoting achievement was attached to the verb *kan4*. Instead, the

[8]Changing the constructional parameters might enable the system to get utterance (3) right, but it would bias it against getting utterance (1) correct.

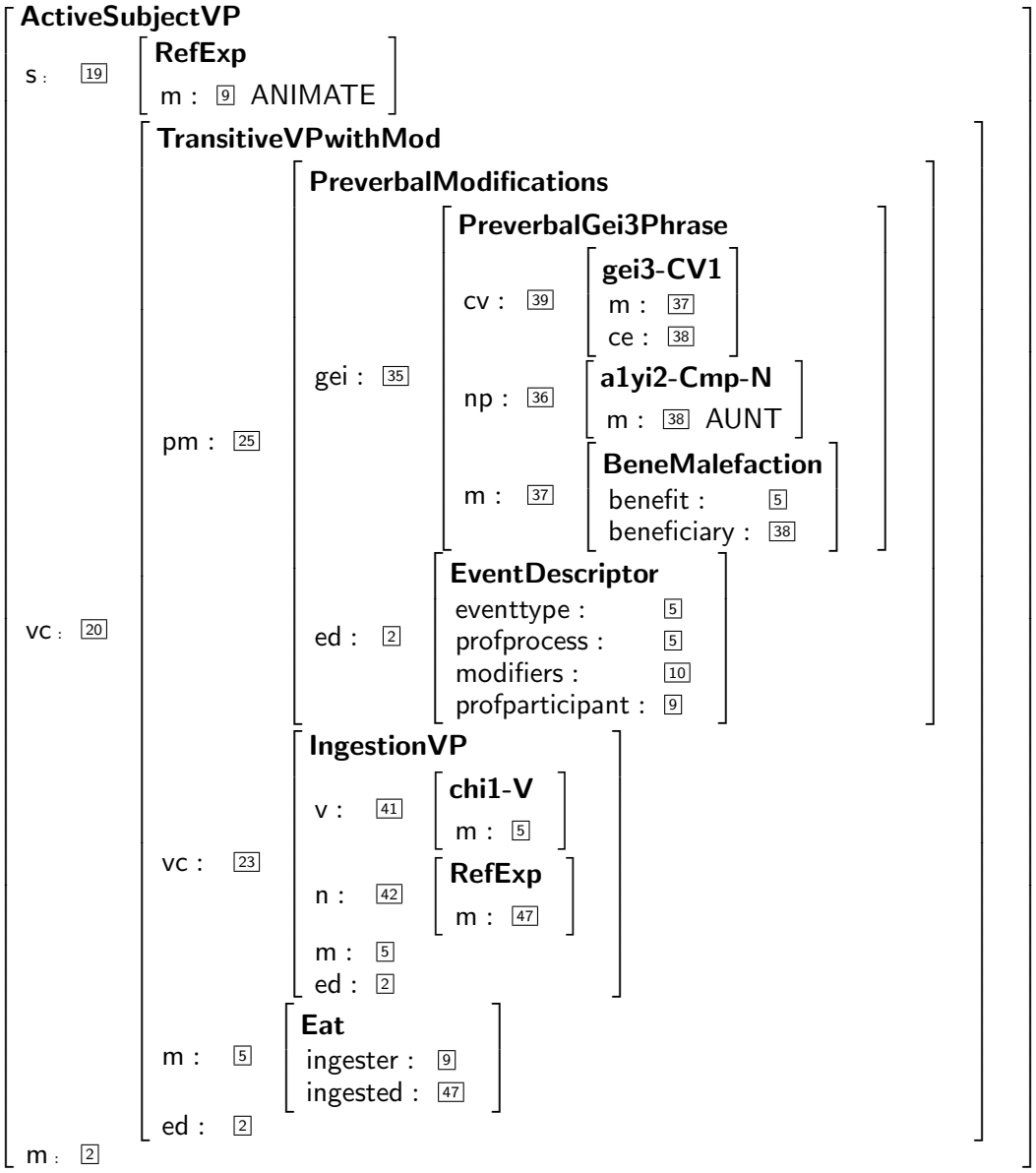

Figure 8.4: The incorrect compressed analysis of utterance (3).

system treats *dao4* as a modifier of *dian4+deng1* meaning *light* (referring to the desk lamp). Figure 8.5 shows the incorrect analysis of utterance (4).

The analysis in figure 8.5 is the first analysis in which I included the DiscourseSegment schema because the particle *ba* is indicative of a request for confirmation (a ResponseEliciting speech act). The FiniteClauseSFP binds the DiscourseSegment's speechAct role to the meaning of its sentence final particle (sfp) role.

Like the other analyses in this section, the system returns the ActiveSubjectVP construction, except this time, the subject is not omitted, and the argument structure construction is PerceptionWithNPArg. Construction PerceptionWithNPArg takes the main verb kan-4 (to see), as its constituent v

The trouble in this analysis is found in the CN construction. The achievement construction dao4-v is bound to the stative modifier (mo) role of the ModifierPlusNoun construction. This turns dao4-v into the modifier of the noun dian4deng1.[9]

The fact that the analyzer incorrectly allows the achievement marker to modify *lamp* illustrates the fact that the best-fit metric must have a notion of what modifications are likely and unlikely for a given referent. [10] To encode the notion of likely and unlikely modifications within the framework of the semantic factor defined in chapter 4, each property (or modification) would have to be reified as a schema, and the modified referent would have to be set as the holder of that property.

8.6 Learning the Parameters

In an ideal world, one would use a maximum likelihood estimate over an "analysis bank" to estimate the constructional and semantic parameters of the factored model. Unfortunately, no such analysis bank exists. The closest thing to an analysis bank that we have access to is the annotation set that Mok built for the training and validation sets, but those annotations do not carry any constructional information. Thus we investigated an EM-style iterative approach to learning the constructional

[9]Adjectives in Mandarin are generally analyzed by linguists as stative verbs. Stative verbs can be used pre-nominally, with or without an associative particle.

[10]As it stands, a bit more semantic information would have to be included in the grammar to make it obvious that the notion of achievement was modifying the referent of the CN construction.

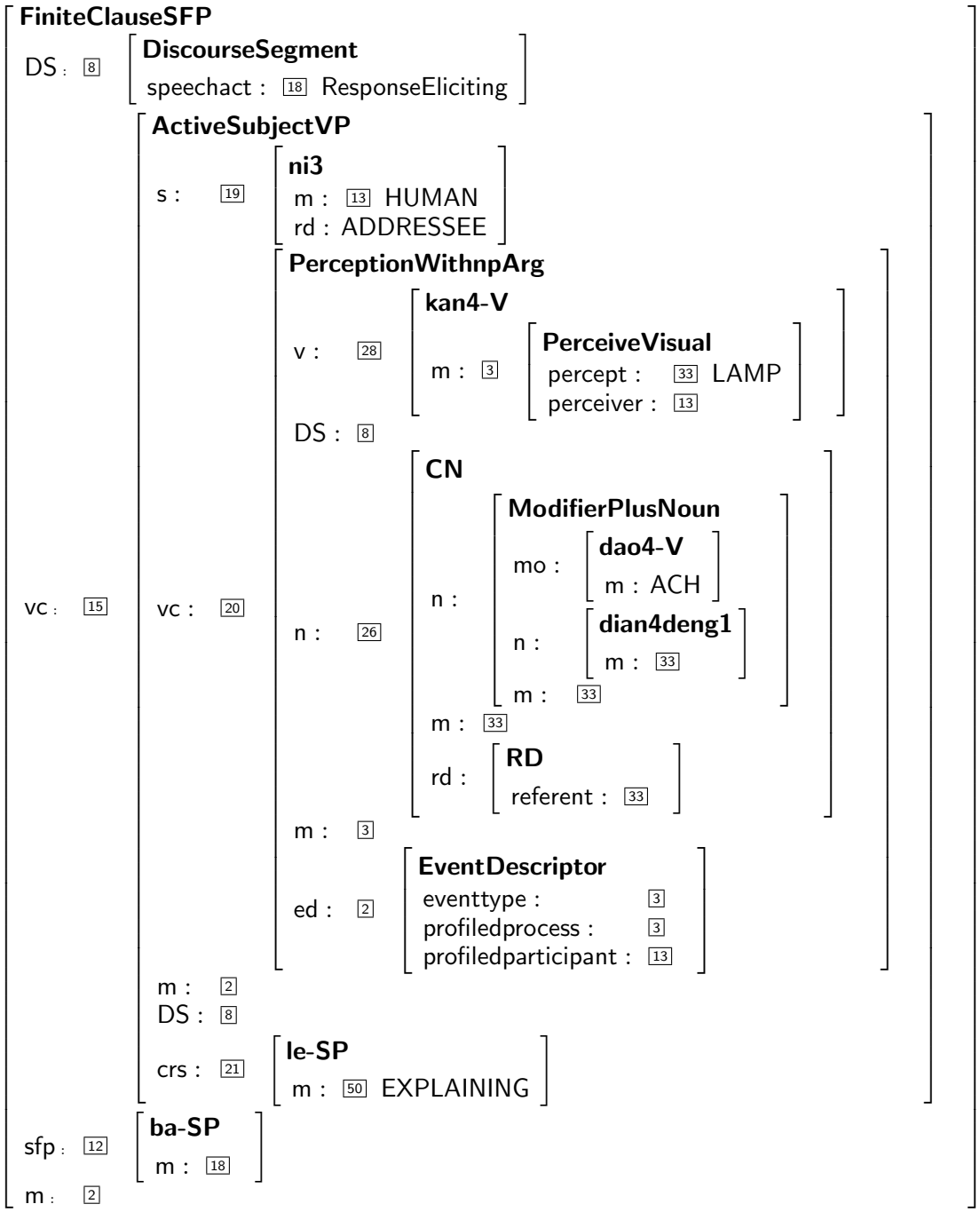

Figure 8.5: The incorrect analysis of utterance (4).

and semantic parameters[11] with alternating parsing and estimation phases:

Input: A corpus θ, a fixed grammar G, and a parameter set ρ consisting of constructional parameters and semantic parameters. For the initial parsing step the constructional filler parameters are assumed to be uniform and the semantic parameters are not used.

Step 1: For each utterance u in corpus θ, parse u using grammar G and parameters ρ

Step 2: Re-rank the analyses of each utterance u according to how well they match u's annotation.[12] The best re-ranked analysis of u is added to a set κ

Step 3: Pretend that κ is a set of gold standard annotations, and re-estimate the constructional and semantic parameters in ρ using a maximum likelihood estimate.

Step 4: Go back to Step 1, using the new parameters in ρ. Repeat until performance on the validation set drops.

On the training set, we only estimated parameters using the 437 utterances for which the grammar produces a single-rooted parse. After each iteration, we used the parameters of ρ and re-parsed the training and validation sets and compared the single best parse to the annotation. Performance was measured by how well each utterance's single best parse predicted the core arguments of the utterance, as defined by the annotation. On the training set, there were 1727 core argument roles to be predicted, and on the validation set there were 444 such roles over the 129 complete parses it found.

Both the constructional parameters and the semantic parameters were smoothed after estimation. The constructional filler parameter Pr(*filler* | *constituent*) was smoothed with Pr(*filler* | *constituent type*) and finally with a uniform distribution

[11]The frame annotations that Mok built for the Childes data are more general than the schemas used in her grammar.

[12]The matching process checked for consistency between core arguments and word senses used.

given the type constraint. The held-out mass was calculated using the Witten-Bell smoothing algorithm on each distribution[35].

The semantic parameters Pr($role = filler \mid frame, filler$) were calculated using a backoff method. For each (frame, filler, role) triple, 1 was added to the counts of each supertype of the filler in addition to the filler itself. For example, if the triple was (Eat, Child, Agent), then (Eat, Child, Agent), (Eat, Person, Agent), (Eat, Animate, Agent), (Eat, ConcreteEntity, Agent), and (Eat, Entity, Agent) would all have their counts incremented. The probability of a new (frame, filler, role) is defined as the normalized count of that triple if it is greater than zero. Otherwise, the probability of the triple is recursively defined as (filler, parent(filler), role).

Table 8.1 shows the performance of the system on the training and validation sets without using semantic parameters on any of the iterations. As can be seen from the table, the iterative technique for estimating the constructional parameters increases precision and recall on both the training and validation sets. Additionally, on the training set, the number of states processed by the parser dropped dramatically from over 31000 with uniform parameters to about 14500 after iteration 2.

Table 8.2 shows the learning progression if the semantic parameters are estimated along with the constructional parameters. Just like in the constructional parameter only case, the precision and recall increase through iteration 1 and then level off at iteration 2. Although the F-scores are quite similar, the precision is higher and the recall is lower when the semantic parameters are included (as one would expect with additional constraints).

8.7 Summary

This chapter shows how the constructional analyzer can be applied to a language with productive omission such as Mandarin. The analyzer uses a context model to resolve the omitted arguments, leveraging the rich semantics of a construction grammar to inform the choice of a referent. Additionally, extensions to the analyzer for parsing with multiple-roots are also defined.

The analyzer was applied to a subset of the Tardiff Beijing Childes Corpus using

Learning without the semantic parameters

Iteration	Training Set			Validation Set		
	Precision	Recall	F-Score	Precision	Recall	F-Score
-1	.811	.688	.744	.805	.669	.731
0	.834	.775	.803	.859	.770	.812
1	.837	.779	.807	.866	.797	.830
2	.837	.778	.807	.865	.797	.830

Table 8.1: The performance of the analyzer on the training and validation sets after each iteration of parameter learning. Iteration -1 is the initialization phase that uses uniform parameters. The semantic parameters were not used during this experiment.

Learning with the semantic parameters

Iteration	Training Set			Validation Set		
	Precision	Recall	F-Score	Precision	Recall	F-Score
-1	.811	.688	.744	.805	.669	.731
0	.836	.763	.798	.870	.768	.816
1	.840	.770	.804	.879	.787	.831
2	.838	.770	.803	.881	.784	.830

Table 8.2: The performance of the analyzer on the training and validation sets after each iteration of parameter learning. Iteration -1 is the initialization phase that uses uniform parameters. The semantic parameters are estimated and used for each iteration except for iteration -1.

a grammar written by Eva Mok. With the grammar, the analyzer could parse about 82% of the utterances in the training set. A closer examination of 125 of the parseable utterances in the corpus reveals that the analyzer returns a perfect match 68% of the time, and how the errors are spread over the remaining 32% of the utterances. Example analyses illustrate how omission is handled by the system as well as showing some of the drawbacks of the current best-fit metric.

The final section of this chapter discusses a pilot study that investigated a machine learning approach to estimating improved constructional and semantic parameters. Using the learned parameters, the F-score on the validation set increases by .1 over the unparameterized grammar. In addition to being more accurate, the system also parses faster with the learned parameters, decreasing the number of states processed by 50%.

Chapter 9

Conclusion and Future Work

This dissertation has shown that a construction-based system for language interpretation, called the constructional analyzer, can be both a cognitive model of interpretation and a practical program for semantic analysis. The analyzer produces rich linguistic analyses with an embodied grammar. It makes incremental reading time predictions consistent with experimental data. It can also robustly parse Mandarin child-directed language, even with Mandarin's dependence on context. The analyzer is the first cognitive model of interpretation that can do all three of these tasks.

The power of the analyzer comes from constructions. Constructions provide explicit constraints between form, meaning and context, thus enabling the use of a best-fit technique for defining the conditional probability of an analysis. Additionally, because the relation between form and meaning is explicit, constructions make it easy to adapt the system to multiple tasks such as the three case studies described in chapters 6, 7 and 8.

The best-fit metric is implemented as a factored probabilistic model over syntax and semantics. The syntactic factor incorporates construction-specific preferences about constituent expression/omission and the kinds of constructional fillers preferred by each constituent. The semantic factor scores a semspec in terms of the fit between roles and fillers. Importantly, the two factors are not treated as independent, but are related through a function that provides the likelihood of the semspec's structure

given a partial constructional tree (or derivation). The function that estimates the probability of a semspec given a derivation makes it possible for the best-fit metric to act as the online search heuristic for the left corner parsing algorithm. Context also affects the search heuristic in a limited way in that referents must be found for all omitted constituents and referring expressions. If no candidates are available, then the analyzer will not pursue an analysis that requires the referent.

The analyzer produces rich linguistic analyses for a range of interesting constructions including embodied semspecs for the various motion and force-application constructions designed by Dodge. An array of syntactically interesting constructions are also easy to implement within the analyzer including constructions for passives, simple wh-questions, raising and radial category description of the ditransitive argument structure construction. Crucially, all of these constructions are implemented within the same internally consistent grammar.

Although the English construction grammar is currently the most linguistically well-motivated grammar processed by the analyzer, the analyzer is not tied to English. It analyzes Mandarin child-directed utterances as well, using a Mandarin grammar. Productive omission is incorporated into the parser and scored by the best-fit metric. Omitted arguments are resolved to a candidate set by the system's context model. The best-fit metric is vital for analyzing the Mandarin data. Even with small biases in the constructional probabilities, the accuracy and efficiency of the system increase dramatically. Success with the Mandarin data confirms that the analyzer is practical enough to be used as a small-scale construction-based semantic analysis engine.

In addition to practicality, analyzer has been designed with a focus on being psychologically and cognitively plausible and predicts differences in incremental reading time for reduced relative data. The factored syntactic and semantic model plays an important role in making the reading time predictions. The syntactic factor implements the syntactic bias for main verb interpretations over reduced relative interpretations[1], and the semantic model implements the good agent/good patient biases that differentiate the two experimental conditions. Crucially, the system is a model

[1]Given just an NP and a past-participle

of deep semantic interpretation first, and it just happens to predict reading time data as a byproduct of its whole design.

9.1 Future Work

Although the constructional analyzer can perform a collection of tasks that no other language interpretation program can, there is still so much left to be done. This section outlines related research directions and for each topic, suggests a roadmap for future work.

9.1.1 More of the Same

For each of the case studies, doing more of the same would strengthen the claims about rich linguistic capability/adequacy, psychological/cognitive plausibility and cross-linguistic completeness. For example, the more argument structure constructions the system can model, the better. The resultative family of constructions would be an ideal test case because it is a well-studied construction with a rich semantic structure. Of course, productivity is also of primary interest, and representing other interesting syntactic phenomenon in ECG will certainly test the adequacy of the both the formalism and the assumptions made in chapter 6.

In regards to psychological plausibility, the McRae et al. data [51] presented in chapter 7 is a tailor-made test case for the constructional analyzer because the data depend on semantics. But the contention that the constructional analyzer is a psychologically plausible model of language interpretation will rest on using the analyzer to model more experiments. For example, the experiments presented by Tabor et al. [81] also depend on the reduced relative, and if their experiments could be modeled, then it would put to rest some of their concerns about mixed parsing strategies. And though the model of context is rudimentary, the constructional analyzer is the only system that has a structured representation of context, and thus it would be the only current system that could potentially model results that condition on contextual information.

The Chinese grammar described in chapter 8 acts as an existence proof that the analyzer can interpret languages that differ significantly from English. Obviously, the more languages that are incorporated, the better. Ongoing work by Nathan Schneider is investigating representing Hebrew within ECG. Semitic languages such as Hebrew have a morphology that is very different from English, and their interpretation would be both scientifically interesting and have obvious applications.

9.1.2 Parameter Learning

One major unsolved problem with the constructional analyzer is how to estimate the constructional and semantic parameters on which the system relies. One promising approach is laid out in chapter 8 for learning the constructional and semantic parameters for the Mandarin grammar. The parameter estimation process specified in chapter 8 could be applied to any corpus for which a construction grammar exists and there was a semantic method for measuring analyzer performance. Even shallow semantic annotation such as FrameNet [22] or Propbank [61] could in principle be used for this process. Of course, one either has to write the construction grammar or a method for inferring the rules of the grammar from data.[2]

9.1.3 Information Extraction

One possible practical application of the system is shallow understanding or information extraction for question answering [75]. Frames and frame role filler extraction of the sort defined by Gildea [24] provide a shallow, but still useful level of semantic representation. The FrameNet project[3] has specified frames for a broad range of lexical constructions. Crucially, they also provide annotations that link the grammatical roles of the annotated utterance to the semantic roles of the evoked frame. In his work on paraphrasing using FrameNet, Michael Ellsworth has shown that these annotations can be construed as extremely impoverished, lexically-specific argument

[2]For more on construction grammar learning see [8] [52].

[3]framenet.icsi.berkeley.edu

structure constructions.[4]

Thus to build a shallow information extraction grammar, one could convert the grammatical information stored inside of FrameNet into shallow ECG constructions and convert the FrameNet frames into ECG schemas. Obviously, additional grammatical scaffolding would have to be added to support noun phrases, conjunction, and different speech acts, but the multi-rooted analysis algorithm could be used if this proved to be too onerous. Then one could use the parameter estimation techniques proposed in chapter eight to infer the constructional filler probabilities needed for the syntactic factor. For the semantic factor, the FrameNet semantic clustering work of Pado [59] could be used as the semantic factor.

9.1.4 Metaphor

For the productive, relatively fixed metaphors[5] covered in [44], there are at least two open questions. The first is how to represent metaphor within ECG, and the second is how metaphor is evoked by constructions. This section provides a brief snapshot of ongoing work for integrating metaphor into the analyzer.

Ongoing work on the representation of productive metaphor in ECG has shown that instances of metaphoric mappings between domains are not so representationally different from schemas. Our current work suggests that metaphors are kinds of maps, and thus a new ECG primitive called *map* has been introduced. Consider the *extremely* simplified LoveIsAJourney metaphor shown in figure 9.1. Like schemas, maps are arranged in a type lattice supporting inheritance. The LoveIsAJourney metaphor is a subcase of a generic Metaphor type, though in a more complete specification, it would be a subcase of the event structure metaphor.

Like all maps, the LoveIsAJourney metaphor has roles defined for a source domain and a target domain. In this case, the source and target are simplified frames for a journey (the JourneyFrame) and a relationship (the RelationshipFrame) which are not shown. A map specifies pairings between the source domain and the target domain.

[4] Personal communication

[5] Methods for dealing with dynamically mapped domains are still an open topic of research and will not be dealt with here.

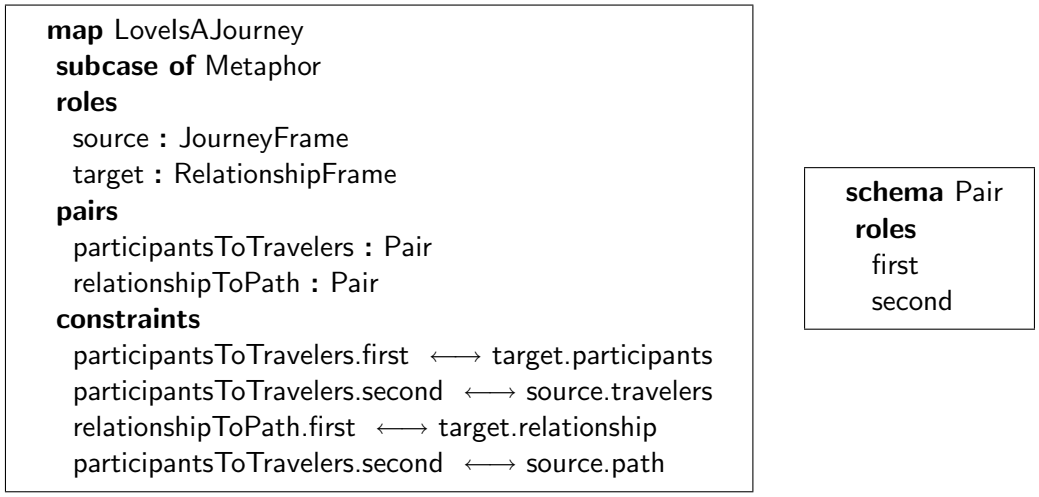

Figure 9.1: An example Love as a Journey metaphor.

In the case of the LoveIsAJourney metaphor shown in figure 9.1, there are two pairings. One pairing is between the participants in the relationship and the travelers of the journey. The other pairing is between the relationship role of the RelationshipFrame and the path role of the JourneyFrame. The constraints block unifies the roles of each pair with the appropriate roles of the source and target domain.[6]

The simplest approach to bringing metaphor into the EJ1 grammar is to define metaphor specific versions of argument structure constructions and prepositional phrase constructions. Consider the utterance[7], *Our relationship is running into trouble.* There are two clues that this utterance is metaphoric. The first is that *relationship* cannot be a literal mover, and the second is the object of the *into* preposition is a state and not a location.

Figure 9.2 shows a simple example metaphorical prepositional phrase construction called StatesAreLocationsPP. It is a subcase of the standard SpatialPP which means it inherits constituents for the preposition, the object of the preposition obj, and the form constraints from SpatialPP. It also inherits the meaning constraints from SpatialPP

[6]While this definition of pairs and constraints might seem redundant, it makes it easy to support static and dynamic maps with the same ECG primitive.

[7]The sequence *relationship is running into trouble* exists on google.

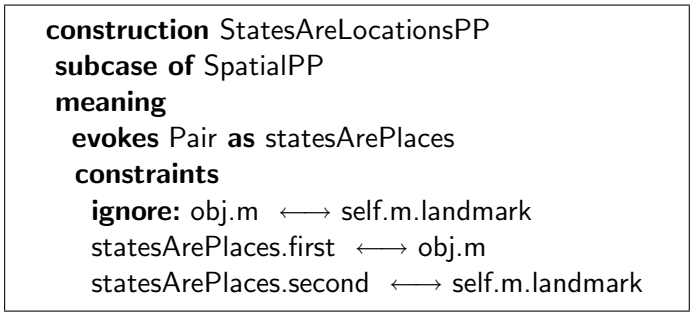

Figure 9.2: A metaphorical PP that takes a state instead of a location.

including typing the meaning pole to TrajectorLandmark and binding the meaning of obj to the ssflandmark role.

All of the differences between SpatialPP and StatesAreLocationsPP are shown in figure 9.2. Importantly, the construction evokes a Pair schema called StatesArePlaces for representing the link between states and places. The constraints block ignores (overrides) the inherited constraint that the meaning of the obj constituent is bound to the landmark role. Instead, the meaning of obj is bound to the first role in the statesArePlaces pair. The landmark role is then bound to the second role of the statesArePlaces pair.

To make the grammar internally consistent, the metaphoric versions of the motion constructions will use StatesAreLocationsPP instead of the standard SpatialPP. The metaphoric motion constructions will then be able to unify the statesArePlaces pair inside the StatesAreLocationsPP with their own statesArePlaces pair.

The approach to metaphor described here might be reasonable for extremely productive constructions like the Event Structure Metaphor which could reasonably be considered reified. The question of what to do for less productive metaphors is still unsolved. Making a separate argument structure construction for every compatible metaphor does not seem like the right answer, so going forward will require more research.

9.1.5 Metonymy

Like metaphor, one can split the concept of metonymy into two subcategories. One kind of metonymy is explicitly grammatically licensed. The other kind of metonymy is a productive use of language for reference.

As an example of the grammatically licensed variety of metonymy, consider the sentence, *the truck roared down the hill*. The sentence features a constructionally specified metonymy usually described as sound for motion. Figure 9.3 shows one possible way to define a sound for motion construction. The SoundForMotion construction inherits from the MotionAlongAPath construction defined in chapter 6 and uses the schemas defined in that chapter as well.

To model the semantics of co-timed motion and sound, construction SoundForMotion evokes a ComplexProcess schema (also defined in chapter 6) called proc and a SoundEmission schema called se. The constraint that unifies the meaning of the verb with the construction's MotionAlongAPath meaning pole is ignored, and instead, the bindings specify that MotionAlongAPath is process1, while the meaning of the verb v.m is unified both with process2 and the SoundEmission schema named se. This ensures that only sound emission verbs can be a part of this construction. The final two constraints bind the protagonist roles appropriately.

The productive kind of metonymy used in referring expressions is exemplified by the utterance, *the ham sandwich wants his check*. For that utterance to make sense, you have to put yourself in the shoes of a waiter that only can distinguish his customers by the food that they ordered, and thus, "ham sandwich" might be the referential description that maximizes both ease of description and ease of reference resolution.

To properly resolve referential descriptions such as these, the approach I advocate is to build the metonymic process into the reference resolution mechanism. My claim is that this kind of metonymy is quite similar to frame-based coreference examples such as the definite use of *the door* in the utterance, *I bought a new car, but the door was scratched*. In this case, *the door* requires inference through the car frame to link *the door*'s meaning with the particular car door.

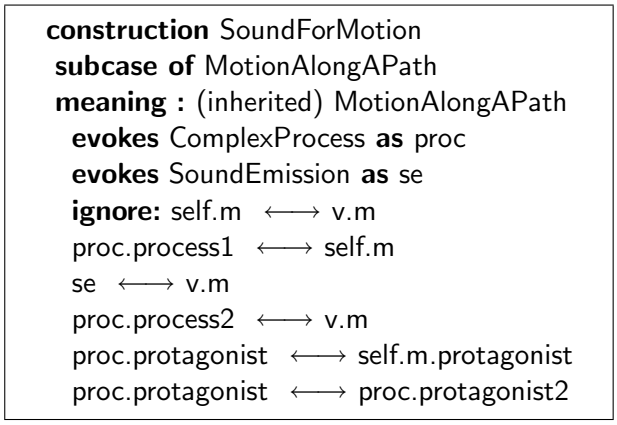

Figure 9.3: The Sound-For-Motion construction and the construction it subcases.

For *the ham sandwich*, a frame-based inference is also necessary, in particular a restaurant patron frame would have a role for the food that she ordered. The motivation for treating *the ham sandwich* as a metonymy (as opposed to simple frame-based coreference) comes from a type clash. The type constraint on *wants* would require a sentient entity, and the ham-sandwich entity would not satisfy such a type constraint. Thus the reference resolution system would instead return the restaurant patron as the appropriate resolved referent.

9.1.6 Morphology, Intonation and Speech Recognition

Morphological analysis, intonation analysis, gesture analysis and speech recognition are separate fields of research. However from the constructional analyzer's point of view, they can all be modeled as co-processes that run in parallel with the analyzer.

- For morphology, linking the constructional analyzer to a morphological analyzer is in progress (joint work with Nathan Schneider). The morphological analyzer takes a morphological grammar that is also construction-based, and returns a set of a lexical tokens for each word of the input string. While the current implementation runs the morphological analyzer on the whole utterance before

phrasal analysis starts, it can still be conceptualized as a co-process that runs in parallel with phrasal analysis.

A bonus for using morphological analyzer is that the morphological analyzer can model the intricacies of building words, and the phrasal analyzer can instead reason about word classes similar to part-of-speech tags. The part-of-speech tags might often be constructionally specific, but regardless, an set of interfaces between the constructional analyzer and the morphological analyzer would reduce the lexical portion of the reachability tables to just the interface types. Of course, semantics and other lexically-specific information will still play a vital role during analysis, but a tag interface makes it possible to collapse the size of the reachability tables to just reachable interface types.

- When processing an acoustic signal, off the shelf speech recognition software can return a list of likely utterances given the acoustic signal. Each of the possible utterances can be processed separately by the analyzer. The likelihood of each utterance's interpretation would have to be scaled by the likelihood of the utterance given the acoustic signal. More precisely, if one wants to infer the most likely interpretation i given acoustic signal α:

$$argmax_i \Pr(\, i \mid \alpha \,) = argmax_i \sum_u \Pr(\, i \mid u \,) \Pr(\, u \mid \alpha \,)$$

  where u ranges over each of the possible utterances given the acoustic signal. Of course, in the actual implementation, one would not enumerate each utterance u, and instead the analyzer would incrementally process the word lattice directly.

- Intonation and gesture carry important information for reference resolution, determining discourse focus, and determining speech act (e.g. sarcasm). Though online gesture processing is still a long ways off, intonation monitoring has been implemented as a co-routine along with speech recognition. An intonation/gesture monitor can also be seen as a co-process running in parallel with the constructional analyzer. Defining an interface between the intonation monitor and the constructional analyzer is subtle, however, since some intonation

boundaries co-occur with word boundaries and some do not. For those that they do co-occur with word boundaries, one possible interface is to add intonational features to morphological rules, and use the morphological analyzer to set the intonational features which would make them available to the phrasal rules when necessary.

9.1.7 Improved Situation and Context Modeling

The context model is an obvious shortcoming of this work. While much of the necessary structure is in the model for doing basic reference resolution, the reference resolution probability model is too rudimentary to be useful. As was addressed in chapter four, commonly used models of reference resolution are not incremental, they are not psychologically plausible, and they are not structured in a cognitively motivated way. Thus even basic research on computationally precise models of context and reference resolution would improve the plausibility of the constructional analyzer.

Though it is not incremental, mental spaces theory [18] provides a cognitively-motivated account of the situational and discourse context that condition language interpretation. Unfortunately mental spaces theory is not computational enough to be employed in the constructional analyzer. Ongoing research with Jerry Feldman and Luca Gilardi is attempting to change this, however.

In the currently evolving account of mental spaces within ECG, each utterance is situated in a *situation* that represents the ongoing discourse and situational context. Different situations are related through *map*s which are quite similar to the metaphor maps described in section 9.1.4. Maps pair elements of different domains. For situations, maps set up counterpart relations that link (for example) person α from a ten year old vacation photograph with the person α of today. These relations enable the cross-situation inference necessary for language understanding.[8]

[8]To continue the vacation photo example, assume that person α depicted in the photograph remarks *I should have brought more sunscreen when I went to Maui.* If the person α is fair-skinned and prone to sunburn, the system can infer that the person depicted in the photo is also prone to sunburn using the map. Only then would the system be able to correctly infer that person α probably did not apply the appropriate amount of sunscreen and got burnt. Of course, such inference is beyond the scope of this dissertation, but in principle, the analyzer could return the necessary

Even with the representational tools of situations and maps, another open problem is how these situations and maps are instantiated. Certain constructions, called *space builders*, are linguistic cues that indicate the possible evocation of a new situation or space. All the problems of incorporating metaphor into the grammar also exist with situations and maps with the additional complication that the maps are dynamic in that new pairs can be introduced by the linguistic content.

Assuming that the situations and maps can be set up properly from the linguistic context, certain referential expressions could refer to either an entity in the current situation, or that entity's counterpart in some other situation. Just as before, the RD schema can still act as the analyzer's interface to context, but with the additional possibility that a resolved referent comes from a different space. For a referring expression that explicitly expresses the fact that it refers to the counterpart of some entity, this fact should be marked on the RD schema. One approach to marking the RD would be to add new RD roles called counterpartOf and baseSituation which would allow the context model to resolve referents in other spaces without the need for extending ECG. As an example, consider the sentence, *The building in the picture has been remodeled.* The NP *the building in the picture* refers to the real-life counterpart of the building in the picture. Using the RD based approach suggested here, the RD might look like:

RD	
category :	building
givenness :	definite
baseSituation :	pictureSituation7
counterpartOf :	pictureSituation7.depictedItem
referent :	building314

In this case, the phrase *the picture* would resolve to pictureSituation7 and fill in the baseSituation role. Use of *in* would indicate the depictedItem of pictureSituation7, thereby filling in the counterPart role. During reference resolution of the whole noun phrase, the context model would search for a counterpart of the item depicted item that was a building, and return building314.

structure to enable the inference.

9.1.8 Generation

Beyond the semantic and constructional factors that are already incorporated into the model, there are a range of additional factors that condition language generation. Besides genre, register and discourse factors (some of which will be covered below), the speaker has to model what the addressee does and does not know. Models of other minds are even beyond the scope of this dissertation's future work section.

However, if one assumes that the necessary factors for conditioning generation are accessible, then one can borrow an idea from statistical machine translation as a starting point for generation [89]. If one assumes that the "source language" is the conceptualization c that needs to be generated, and that the "target language" is an utterance u, one can mathematically define the process of generating u as:

$$argmax_u \Pr(\, u \mid c \,)$$

Then using Bayes rule to rewrite $\Pr(\, u \mid c \,)$:

$$argmax_u \Pr(\, u \mid c \,) = argmax_u \Pr(\, c \mid u \,) \Pr(u)$$

Term $\Pr(\, c \mid u \,)$ is the likelihood of the conceptualization given the utterance. This term models the faithfulness of the utterances to the conceptualization c, penalizing utterances that are ambiguous. Term $\Pr(u)$ is the prior likelihood of the utterance, and it measures the fluency of the utterance u. Highly likely utterances are generally easier to process. An utterance u that maximizes the product of $\Pr(\, c \mid u \,)$ and $\Pr(u)$ would have to be both easy to process and faithfully generate conceptualization c. Notice that $\Pr(\, c \mid u \,)$ is basically the probability of an analysis given a sentence, which is what the analyzer estimates when it parses an utterance. Additionally, it is straightforward to estimate the likelihood of the utterance u using the analyzer.

Just as in the decoding process of machine translation, one must define a way of generating possible candidate utterances in the "target language". These candidate utterances act as the domain of $\Pr(u)$. Assuming that the conceptualization c was represented as an EventDescriptor schema[9] with bindings for the profiledParticipant,

[9] If this is not the case, then another process that maps the conceptualization c onto an EventDescriptor is also needed.

profiledProcess, and topic roles, then one would hope that the decoding process and the analysis process could be performed and modeled jointly. Abstractly, the process would be kick-started by generating the topic of the sentence, and then choosing to generate a likely word given the prefix, but only if the generation of that word resulted in a semspec that was still unifiable with the original c. To guide the search for candidate utterances, an A^* heuristic indicating how many remaining unexpressed semantic bindings could be used to indicate how close an utterance u was to matching the conceptualization c.

9.1.9 Simple Ways to Condition the Analysis Process on Discourse, Genre, Register and Priming

One obvious way to condition the analysis process on discourse facts, genre and register is to use hard constraints such as features. Such an approach is not problematic given the unification grammar backbone that ECG is built around. However, softer constraints such as preferences on construction choice in certain discourse contexts, genre and register require extension to the probabilistic model.

One simple way to extend the model is to incorporate "probabilistic switches". This is implicitly what I do for conditioning the likelihood of nonlocal constituents. A probabilistic switch (or more accurately a simple piece of conditioning information) like "existingFrontedItem=true" modifies the conditional likelihood of a nonlocal constituent in a functional (on/off) way. In essence, the probability of a nonlocal constituent is zero unless "existingFrontedItem" is true.

Other kinds of probabilistic switches might be the facts about the discourse such as the type of the previous speech act, the genre of the discourse, and the intended register of the discourse. A simple switch like "PreviousSpeechAct=Wh-Question" would increase the likelihood that the response could be a simple NP instead of a finite clause.

Priming could also be a case where a probabilistic switch could be employed. To do this, the constituent filler probabilities would need to be updated in some simple way if a construction that was primed in the previous utterance were a possibility.

Any modification of the distribution could be used as long as the resulting distribution still summed to one.

9.1.10 A Neural Model

Cognitive and psychological plausibility were of central importance in the design of the constructional analyzer. Though it is not possible to map the analyzer's processes onto actual neural circuitry in the brain, one can use the design of the analyzer to inform the design of a computational abstraction over neural circuitry such as a structured connectionist model. The cognitively/psychologically plausible aspects of the system include:

- Incremental processing
- Conditioning decisions on syntactic and semantic information (i.e. best-fit)
- Limited parallelism and memory
- Top-down and bottom-up influence on parsing
- Deep semantics
- Constructions

Lane and Henderson use a connectionist model of a best-fit function in their left corner parsing work [45]. It is a recurrent network that models the relevant features of the current derivation, and it predicts the likelihood of each next derivational step. They use a specialized recurrent network that contains co-timed "pulsating" nodes to represent the structure of a parse and train it using backpropagation.

More cognitively plausible approaches to representing structured elements are devised by Shastri et al. [74] using temporal synchrony and the generalization over temporal synchrony proposed by Barrett et al. [2] which supports unification. Assuming that the Barrett et al. work could be extended to support a limited stack-like item as one of their model's variables, then each of the variables could represent competing interpretations of an utterance. The new structure associated with each

word would be unified in with each of the stacks and unlikely resulting stacks would be pruned away because of lack of activation. The stack with the highest activation that survives to the end of the utterance would be the best-fit interpretation of the utterance.

9.2 Summary

As you can see, best-fit constructional analysis lays the groundwork for computational implementations of cognitively motivated theories of morphology, metonymy, metaphor, generation, and mental spaces. Additional practical applications include tasks where rich semantic analysis is necessary such as question processing or dialogue systems. And a connectionist implementation of the system would not only advance the field of connectionist modeling, but cognitive science as well.

Bibliography

[1] Steven Abney. Stochastic attribute-value grammars. *Computational Linguistics*, 1997.

[2] Leon Barrett, Jerome Feldman, and Liam Mac Dermed. A (somewhat) new solution to the variable binding problem. *Neural Computation*, 2008.

[3] Benjamin Bergen and Nancy Chang. *Embodied Construction Grammar in Simulation-Based Language Understanding*. John Benjamins, 2007.

[4] Chris Brew. Stochastic hpsg. In *Proceedings of 7th Conference of the EACL*, 1995.

[5] John Bryant. Constructional analysis. Master's thesis, UC Berkeley, 2003.

[6] Sharon A. Caraballo and Eugene Charniak. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298, 1998.

[7] Combinatory categorial grammar site: groups.inf.ed.ac.uk/ccg/index.html.

[8] Nancy Chang. *Forthcoming... (Argument Structure Construction Learning)*. PhD thesis, University of California at Berkeley, 2008.

[9] Nancy Chang and Tiago Maia. Learning grammatical constructions. In *Proceedings of the Conference of the Cognitive Science Society*, 2001.

[10] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational*

Linguistics, pages 132–139, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[11] Michael Collins. Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL*, 1997.

[12] Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 2005.

[13] Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm.

[14] Peter Culicover and Ray Jackendoff. *Simpler Syntax*. Oxford University Press, 2005.

[15] Alan Demers. Generalized left corner parsing. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Programming Languages*, 1977.

[16] Ellen Dodge. *As of yet untitled*. PhD thesis, University of California at Berkeley, 2008.

[17] Jay Earley. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, 1970.

[18] Gilles Fauconnier. *Mappings in Thought and Language*. Cambridge University Press, 1997.

[19] Jerome Feldman. *From Molecule to Metaphor: A Neural Theory of Language*. MIT Press, 2006.

[20] Charles Fillmore. Frame semantics. In *Linguistics in the Morning Calm*, pages 111–138. Linguistics Society of Korea, 1982.

[21] Charles Fillmore, Paul Kay, and M. C. O'Connor. Regularity and idiomaticity in grammatical constructions: the case of *let alone*. *Language*, 64(3):501–538, 1988.

[22] Charles J. Fillmore and Collin F. Baker. FrameNet: Frame semantics meets the corpus. In *Linguistic Society of America*, January 2000.

[23] R. Ge and Ray Mooney. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the 9th Conference on Computational Natural Language Learning*, 2005.

[24] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

[25] Jonathan Ginzburg and Ivan Sag. *Interrogative Investigations: The Form, Meaning, and Use of English Interrogatives*. CSLI, 2001.

[26] Adele Goldberg. *Constructions: A Construction Grammar Approach to Argument Structure*. University of Chicago Press, 1995.

[27] Aria Haghighi, John DeNero, and Dan Klein. Approximate factoring for a* search. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 412–419, Rochester, New York, April 2007. Association for Computational Linguistics.

[28] Aria Haghighi and Dan Klein. Unsupervised coreference resolution in a nonparametric bayesian model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 848–855, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[29] John Hale. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of NAACL*, 2001.

[30] John Hale. Uncertainty about the rest of the sentence. *Cognitive Science*, 2006.

[31] James Henderson. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Meeting of the ACL*, 2004.

[32] James Henderson. Lookahead in deterministic left-corner parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, 2004.

[33] Mark Johnson, Stuary Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. Estimators for stochastic unification-based grammars. In *Proceedings of the ACL*, 1999.

[34] Daniel Jurafsky. A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, 20:137–194, 1996.

[35] Daniel Jurafsky and James Martin. *Speech and Language Processing*. Prentice Hall, 2000.

[36] Hans Kamp and Uwe Reyle. *From Discourse to Logic - An Introduction to the Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer, 1990.

[37] Paul Kay. An informal sketch of a formal architecture for construction grammar. *Grammars 5*, 2002.

[38] Paul Kay and Charles Fillmore. Grammatical constructions and linguistic generalizations: the what's x doing y? construction. *Language*, 1999.

[39] Dan Klein and Christopher D. Manning. A* parsing: Fast exact viterbi parse selection. Technical Report dbpubs/2002-16, Stanford University, 2001.

[40] Dan Klein and Christopher D. Manning. A* parsing: Fast exact viterbi parse selection. In *Proceedings of HLT-NAACL 03*, 2003.

[41] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 2003.

[42] Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In Suzanna Becker, Sebastian Thrun, and

Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

[43] George Lakoff. *Women, Fire, and Dangerous Things.* University of Chicago Press, 1987.

[44] George Lakoff and Mark Johnson. *Metaphors We Live By.* University of Chicago Press, 1980.

[45] Peter Lane and James Henderson. Incremental syntactic parsing of natural language corpora with simple synchrony networks. *IEEE Transactions on Knowledge and Data Engineering*, 2001.

[46] Roger Levy. *Probabilistic Models of word order and syntactic discontinuity.* PhD thesis, Stanford, 2005.

[47] Lexical functional grammar site: www.essex.ac.uk/linguistics/lfg/.

[48] Chris Manning and Bob Carpenter. Probabilistic parsing using left corner language models. In *Proceedings of the 5th International Workshop on Parsing Technology*, 1997.

[49] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1994.

[50] Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to noun coreference. In *Proceedings of Neural Information Processing Systems*, 2004.

[51] Ken McRae, Michael Spivey-Knowlton, and Michael Tannenhaus. Modeling the effect of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38:1283–312, 1998.

[52] Eva Mok. *Forthcoming... (Mandarin Grammar Learning with Omission).* PhD thesis, University of California at Berkeley, 2008.

[53] Eva Mok and John Bryant. A best-fit approach to productive omission of arguments. In *The 32nd Annual Meeting of the Berkeley Linguistics Society*, 2006.

[54] Srini Narayanan. *Knowledge-Based Action Representations for Metaphor and Aspect*. PhD thesis, University of California at Berkeley, 1997.

[55] Srini Narayanan and Dan Jurafsky. A bayesian model of human sentence processing. in preparation.

[56] Srini Narayanan and Dan Jurafsky. A bayesian model predicts human parse preference and reading time in sentence processing. In *Advances in Neural Processing Systems 14*, pages 59–65. MIT Press, 2002.

[57] Srini Narayanan and Daniel Jurafsky. Bayesian models of sentence processing. In *Proceedings of the Conference of the Cognitive Science Society*, 1998.

[58] Vincent Ng and Claire Cardie. Improving machine learning approaches to coreference resolution. In *Proceedings of 40th Conference of the ACL*, 2002.

[59] Ulrike Pado. *The Integration of Syntax and Semantic Plausibility in a Wide-Coverage Model of Human Sentence Processing*. PhD thesis, Saarland University, 2007.

[60] Ulrike Pado, Frank Keller, and Matthew Crocker. Combining syntax and thematic fit in a probabilistic model of sentence processing. In *COGSCI06*, 2006.

[61] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

[62] Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April 2007. Association for Computational Linguistics.

[63] Carl Pollard and Ivan Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994.

[64] Sameer Pradhan, Wayne Ward, and James H. Martin. Torwards robust semantic role labeling. *Computational Linguistics*, 2008.

[65] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. Technical report, University of Pennsylvania, 1997.

[66] Stefan Riezler, Tracy Kind, Ronald Kaplan, Richard Crouch, John Maxwell, and Mark Johnson. Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. 2002.

[67] Stefan Riezler, Detlef Prescher, Jonas Kuhn, and Mark Johnson. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and em. In *Proceedings of the 38th Annual Meeting of the ACL*, 2000.

[68] Brian Roark. *Robust Probabilistic Predictive Syntactic Processing: Motivations, Models, and Applications*. PhD thesis, Department of Cognitive and Linguistic Sciences, Brown University, 2001.

[69] Carolyn Penstein Rose and Alon Lavie. *Robustness in Language and Speech Technology*, chapter Balancing Robustness and Efficiency. Kluwer Academic Publishers, 2001.

[70] S. Rosenkrantz and P. Lewis. Deterministic left corner parsing. In *IEEE Conference Record of the 11th Annual Symposium on Switching and Automata*, 1970.

[71] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[72] Kenji Sagae, Brian MacWhinney, and Alon Lavie. Automatic parsing of parental verbal input. *Behavior Research Methods, Instruments and Computers*, 2004.

[73] Li Shaozi, Li Wang, and Zhou Changle. Research on chinese ellipsis recovering based on discourse representation theory. In *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence*, 2005.

[74] L Shastri and V Ajjanagadde. From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using termporal synchrony. *Behavioral and Brain Sciences*, 1993.

[75] Steve Sinha. *Forthcoming... (Question Answering with Frames)*. PhD thesis, University of California Berkeley, 2008.

[76] Paul Smolensky and Geraldine Legendre. *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*. MIT Press, 2006.

[77] W Soon and D. Lim H. Ng. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 2001.

[78] Edward Stabler. The finite connectivity of linguistic structure. In Rayner Clifton, Frazier, editor, *Perspectives on Sentence Processing*. Lawrence Erlbaum, 1994.

[79] Luc Steels and Joachim De Beule. A (very) brief introduction to fluid construction grammar. In *Proceedings of the 3rd International Workshop on Scalable Natural Language Understanding*, 2006.

[80] Andreas Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 1995.

[81] W. Tabor, B. Galantucci, and D. Richardson. Effects of merely local syntactic coherence on sentence processing. *Journal of Memory and Language*, 2004.

[82] Leonard Talmy. The fundamental system of spatial schemas in language. 2005.

[83] Twila Tardiff. Adult-to-child speech and language acquisition in mandarin. *Developmental Psychology*, 1993.

[84] Ivan Titov and James Henderson. Constituent parsing with incremental sigmoid belief networks. In *Proceedings of the 45th Meeting of the ACL*, 2007.

[85] Michael Tomasello. *Constructing a Language: A usage-based Theory of Language Acquisition*. Harvard University Press, 2003.

[86] Noriko Tomuro. *Left-Corner Parsing Algorithm for Unification Grammars*. PhD thesis, Depaul University, 1999.

[87] Kristina Toutanova, Christopher Manning, Dan Flickinger, and Stephan Oepen. Stochastic hpsg parse disambiguation using the redwoods corpus. *Journal of Logic and Computation*, 2005.

[88] Dong Hoon Van Uytsel. *Probabilistic Language Modeling with Left-Corner Parsing*. PhD thesis, Katholeike Universiteit Leuven, 2003.

[89] Yuk Wah Wong and Raymond J. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the ACL*, 2007.

[90] Luke Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005.

[91] Luke Zettlemoyer and Michael Collins. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007.