THE UNIVERSITY *of* EDINBURGH

Learning Words and Syntactic Cues in Highly Ambiguous Contexts

Bevan Keeley Jones

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2015

Lay Summary

The problem of how children learn to associate meanings with words in the early stages of the acquisition of their native language is a difficult one, and psychologists have proposed many mechanisms for contending with this challenge, often relying on notions of inborn knowledge or other constraints that make the problem more manageable. However, it remains unclear how necessary this inborn knowledge really is, or, assuming it is necessary, from where the necessity derives. In particular, there are two possible reasons for such a need: (1) limitations on the child's mental resources such as memory or processing power or, alternatively, (2) a possible lack in the information a child has access to from his environment (separate from any innate knowledge he may or may not have).

In this thesis, we explore the fundamental learning problem itself, abstracting away from whatever resource limitations a child may be subject to by using computer simulations of the child's situation. We find that a computer is capable of learning even with very limited innate knowledge, arguing against the idea that children receive too little external information and suggesting rather that perhaps limitations on human mental resources offer stronger support for theories of innate knowledge.

Additionally, there are several valuable novel byproducts of the research, including a data set for testing alternative models and a computational system that can automatically interpret the meanings of sentences. Furthermore, the computational system is built on a very general framework that can be applied to many other problems in natural language processing.

Abstract

The cross-situational word learning paradigm argues that word meanings can be approximated by word-object associations, computed from co-occurrence statistics between words and entities in the world. Lexicon acquisition involves simultaneously guessing (1) which objects are being talked about (the ”meaning”) and (2) which words relate to those objects. However, most modeling work focuses on acquiring meanings for isolated words, largely neglecting relationships between words or physical entities, which can play an important role in learning.

Semantic parsing, on the other hand, aims to learn a mapping between entire utterances and compositional meaning representations where such relations are central. The focus is the mapping between meaning and words, while utterance meanings are treated as observed quantities.

Here, we extend the joint inference problem of word learning to account for compositional meanings by incorporating a semantic parsing model for relating utterances to non-linguistic context. Integrating semantic parsing and word learning permits us to explore the impact of word-word and concept-concept relations.

The result is a joint-inference problem inherited from the word learning setting where we must simultaneously learn utterance-level and individual word meanings, only now we also contend with the many possible relationships between concepts in the meaning and words in the sentence. To simplify design, we factorize the model into separate modules, one for each of the world, the meaning, and the words, and merge them into a single synchronous grammar for joint inference.

There are three main contributions. First, we introduce a novel word learning model and accompanying semantic parser. Second, we produce a corpus which allows us to demonstrate the importance of structure in word learning. Finally, we also present a number of technical innovations required for implementing such a model.

Acknowledgements

There is a hidden army of supporters behind this document, from family and friends to lab mates and other colleagues, without whom the final product would look very different (especially considering it would undoubtedly not exist at all). To list every deserving person's name would be an arduous task, and reading it even more so, but there are a few that deserve special mention.

I thank Benjamin Börschinger and Vasfiye Geçkin for translating the German and Turkish sections of the Frog Stories corpus into English. Without them my work would have been limited to the English-only section and work would have been far too simple and would have only taken one third the time to complete.

I would further like to acknowledge the generous contributions of time and effort of Joel Lang, Michael Auli, Stella Frank, Prachya Boonkwan, Christos Christodoulopoulos, and Ioannis Konstas in providing translations of GeoQuery880 and also Tom Kwiatkowski for organizing and overseeing said translation work. The semantic parsing community owes them their gratitude. My own contribution has been far overestimated with regard to this corpus when in reality I hardly did more than package and upload it to my website.

Kevin Knight and David Chiang also deserve much thanks, for without them it is unclear how much longer it would have taken for me to stumble upon all the abstracts puzzles and games residing in the land of formal language theory. After just one eventful summer, I now seem doomed to forever chase a seemingly endless trail of fascinating questions, some of which have already distracted me far more than my advisors would have likely preferred on my path to the product now sitting before the reader.

Finally, it goes without saying that Mark Johnson and Sharon Goldwater deserve much thanks for accompanying me along this journey as my mentors. Obviously, this dissertation would not be in the state it is now (i.e., finished) without their assistance and advice over the years. The remaining flaws are clearly my own, but it is much improved for their counsel and feedback.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Bevan Keeley Jones*)

Table of Contents

Chapter 1

Introduction

Given a visual scene and an utterance describing it, unless one knows what the words mean, it is not at all obvious to which aspects of the scene the utterance may be referring. For instance, in a scene with a boy and his dog sitting alone in a bedroom, an utterance may focus on the boy, the dog, the act of sitting, or any other aspect of the scene. Words may refer to relations between entities as well, describing, for instance, the proximity of the dog to the boy. Additionally, each element of the scene may be associated with any number of concepts, each of which constitutes a possible meaning candidate for a word. The boy may be referred to as "the master" and the dog as "his pet", even though these noun phrases do not specifically denote the concepts of boy and dog, but more abstract ideas involving domesticated animals and their keepers. The sheer number of candidate meanings for even a single word, much less the utterance as a whole, poses a daunting challenge that only grows worse when we consider meanings for entire clauses and sentences. In fact, the problem is related to the challenge of induction pointed out by Quine (1960): specifically, there is always an infinite number of hypothesis that might explain the learner's experience. The scale of the problem has led psycholinguists to posit a number of heuristics by which learners may rapidly narrow the set of possible hypotheses to something much more obviously tractable.

Some argue that learners may rely on a set of simple constraints or biases to prune the space of possible meanings. These constraints may be innate or acquired through experience. They may be specific to language learning or they may arise out of more general properties of cognition. In any event, these constraints do appear to be useful, however they arise and whatever specific shape they take.

Many of the problems of word learning are related to more general issues about

category learning. How does one learn from a potentially very small set of examples that the word “dog” can refer to a chihuahua or to a Great Dane, but not to another four legged furred animal like a cat, or to a body part such as the dog’s tail? To handle these sorts of challenges, psychologists have proposed mechanisms such as the whole object (Carey, 1978; Mervis, 1987), shape (Markman and Wachtel, 1988), or taxonomic bias (Markman, 1989) which may drive learners to prefer categories of certain types over others. These lines of inquiry, while interesting and challenging, are outside the scope of our concern here.

We primarily address another set of problems which have less to do with categories and more to do with the question of how a learner determines which objects are being talked about. One learning constraint that has been proposed to help deal with this referential ambiguity is the principle of *mutual exclusivity*, the idea that the same word is unlikely to be used for two different concepts or for the same concept to be referred to by completely different words (Markman and Wachtel, 1988). This principle has been used to explain how children sometimes seem to learn a word with very little exposure, sometimes on their very first encounter, a phenomenon sometimes referred to as *fast mapping*. The proposal is that children assume that novel words are more likely to map to new concepts, a corollary of the mutual exclusivity principle.

Psychologists have also suggested a number of cues that may assist the learner, sometimes arguing that these can alleviate the need for specific learning constraints.

- **Cross Situational Consistency**: Consistent co-occurrence between certain words and objects in the non-linguistic context can help identify word-object mappings (Pinker, 1989; Gleitman, 1990). For instance, a learner can exploit the situation where hearing the word “dog” seems to increase the probability of there being a dog nearby and vice versa.

- **Salience**: Learners may come to associate the concepts that are most central or salient to a particular scene with the words they hear most frequently at the time (Smith, 2000b).

- **Joint attention and other social cues**: Communicators make an effort to increase the salience of the subject of discussion with gestures, eye gaze, or other cues (Baldwin, 1993). Speakers also increase the salience of words, not just objects in the non-linguistic context, using acoustic qualities of the speech such as pauses, stress, and pitch. (Fernald, 1985).

- **Syntactic bootstrapping**: Knowledge of syntactic features of the language may also assist in learning the meanings of novel words (Gleitman, 1990). For instance, knowing that agents of actions tend to occur earlier in English sentences than the objects being acted upon can help to focus attention on where the word that refers to the boy is likely to occur in a sentence describing the action of the boy petting the dog.

There is a great deal of debate about how much of a role these heuristics and cues play in the word learning task and about how they may interact with one another. For instance, some argue that salience may play such an important role that learners may not need to rely so heavily on other cues or constraints (Smith, 2000b). Similarly, citing the notion of "poverty of the stimulus", some stress the importance of innate constraints over any input the child may receive (Chomsky, 1980). In either of these cases, the learning problem tends to be downplayed, and may be relatively trivial, at least in the early stages. Proponents of the cross-situational paradigm argue that a sophisticated learning mechanism could serve to alleviate some of the reliance on innate constraints and biases, instead exploiting what may appear to be relatively weak cues in the form of co-occurrence statistics (Yu and Smith, 2007).

It seems that the truth of the matter may be some combination of factors. Innate constraints may inform a powerful learning mechanism capable of exploiting many different cues in the data. The question in such a case is not whether these different features are useful but what is the magnitude of their individual contribution and how might they interact in an inclusive theory of language learning? The objective of this dissertation is to explore these questions in the context of a computational cognitive model which can help shed light, if not on precisely how humans actually acquire language, at least on what is possible when a powerful learning mechanism is applied to data. We suspect statistical learning is often underestimated in psychological studies, but this mechanism does not stand alone, and nature, the ultimate opportunist, does not hesitate to exploit whatever features or biases that prove helpful.

Indeed, there are so many different biases and cues which might play a role that it is necessary to restrict consideration to the interaction between just a few. Mutual exclusivity, which is closely related to the concept of sparsity in machine learning, can be explored as a soft constraint in a Bayesian framework with a sparse prior. Additionally, there are other constraints from linguistic theory that have more to do with structural properties of meaning representations, such as the semantic uniqueness, completeness, and coherence conditions for well-formedness in Lexical-Function Grammar (Bres-

nan, 2001), aspects of which we incorporate into our experiments. For the most part, however, the focus is on the contribution of different cues in concert and the ability of a model to effectively exploit them to successfully negotiate a large search space. In particular, we explore the interaction between all four of the cues listed above: cross-situational statistics, salience, social cues, and syntactic information, with a particular emphasis on the last.

Syntactic bootstrapping is often thought to come into play somewhat late in word learning, only after acquiring a basic starting vocabulary. The combination of previously learned words and a few basic facts about syntax can play a significant role in constraining the meanings of any as yet unknown words in a sentence. As the learner's vocabulary grows, such syntactic cues also grow in importance, to the point where syntax can eventually dominate the learning problem. Thus, there is a question of how big must the vocabulary be before syntax begins to win out over other aspects of the problem, and when does it start to become useful at all? In terms of the role of syntax versus other cues, we also ask the question of how constrained must the learning problem be before we begin observing syntactic bootstrapping-like effects?

Largely independent of the psycholinguistic work, there have also been a number of developments related to syntactic bootstrapping in the sub-field of computational linguistics known as semantic parsing. For instance, Börschinger et al. (2011) present a PCFG-based semantic parsing model that can learn a mapping from sentence to a formal representation of its meaning under an assumption of a modest degree of referential ambiguity. Such a semantic parser can be interpreted as a kind of word learning model, and they showed how word learning performance can be improved by simultaneously learning a simple syntactic feature such as the canonical order among the subject, verb, and object of a sentence. In a similar vein, Kwiatkowski et al. (2012) present a joint semantic and syntactic model based on Combinatory Categorial Grammar (CCG) and employ an online learning procedure to similarly learn word order information and argue that such a model can exploit syntactic information to perform one-shot learning, or "fast mapping".

These computational models touch on questions central to our focus here. That is, they each ask the question, "what is the role of syntax in word learning given a powerful learning mechanism?" However, these previous approaches have only begun to address the dynamic relationship among the various constraints and cues posited by psycholinguistics. In particular, such studies have been limited to cases where there is relatively little referential ambiguity, due largely to the computational constraints

of the models and algorithms employed. For instance, Börschinger et al. (2011) experiment in a setting where there are on average about two meaning candidates per sentence and Kwiatkowski et al. (2012) increases this to seven. However, these numbers are in marked contrast to the fully unconstrained setting with its possibly infinite number of candidate meanings, and we show for one corpus in Chapter 7 that even under several simplifying assumptions this number can well exceed 1,000 candidate meanings. Effectively, previous work has operated under an implicit assumption that the vast majority of the ambiguity has already been pre-resolved by some unspecified means. Of course, it is entirely possible that this is a reasonable assumption, i.e., that some combination of cognitive biases and learning cues truly do eliminate much of the ambiguity before syntax begins to play a role. Plausible or not, however, it remains unclear whether this assumption actually is a fundamental necessity or if it is only a limitation of the computational techniques employed. It may be possible that, in the absence of such computational constraints, a learning algorithm could perform much of the disambiguation itself in parallel with the syntactic learning, and that word-to-word and concept-to-concept relations in the syntax and semantics of utterances may even play a crucial role in this disambiguation.

To better understand the problem, consider this logical representation of a very simple scene where there is a boy and a frog looking at each other and the frog is inside a jar:

$$\begin{aligned}\exists e_1, e_2.\,&\text{look}(e_1) \wedge \text{experiencer}(e, x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e_1, x_2) \wedge \text{frog}(x_2) \\ &\wedge \text{loc-inside}(e_1, x_3) \wedge \text{jar}(x_3) \\ &\wedge \text{look}(e_2) \wedge \text{experiencer}(e_2, x_2) \wedge \text{theme}(e_2, x_1) \\ &\wedge \text{loc-inside}(x_2, x_3)\end{aligned}$$

Suppose that while observing the scene, the learner also hears the utterance:

The boy is looking at the frog.

With no other information besides the scene description and the words of the utterance, the learner must somehow identify what is being described. This scene is quite simple with only three entities (*boy*, *frog*, and *jar*), two different *look* events, and six binary relations, and is is important to note that these numbers can in actual practice be far larger (something that will be argued via a corpus analysis presented in Chapter 7 where scenes contain on average about 27 entities, 46 events, and 114 binary relations).

Even with such a small scene, however, there are already many candidates. Possible meaning candidates include:

$\exists e_1.\text{look}(e_1) \wedge \text{experiencer}(e_1, x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e_1, x_2) \wedge \text{frog}(x_2)$
the boy looked at the frog

$\exists e_1.\text{look}(e_1) \wedge \text{experiencer}(e_1, x_1) \wedge \text{boy}(x_1) \wedge \text{loc-inside}(e_1, x_3) \wedge \text{jar}(x_3)$
the boy looked inside the jar

$\exists e_1.\text{look}(e_1) \wedge \text{experiencer}(e_1, x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e_1, x_2) \wedge \text{frog}(x_2)$
$\wedge \text{loc-inside}(e_1, x_3) \wedge \text{jar}(x_3)$
the boy looked inside the jar at the frog

$\exists e_1.\text{look}(e_1) \wedge \text{experiencer}(e_1, x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e_1, x_2) \wedge \text{frog}(x_2)$
$\wedge \text{loc-inside}(x_2, x_3) \wedge \text{jar}(x_3)$
the boy looked at the frog inside the jar

$\exists e_1.\text{look}(e_1) \wedge \text{experiencer}(e, x_1) \wedge \text{boy}(x_1) \wedge \text{loc-inside}(e_1, x_3) \wedge \text{jar}(x_3)$
$\wedge \text{loc-inside}(x_2, x_3) \wedge \text{frog}(x_2)$
the boy looked in the jar that the frog was inside

$\exists e_1.\text{look}(e_1) \wedge \text{experiencer}(e_1, x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e_1, x_2) \wedge \text{frog}(x_2)$
$\wedge \text{loc-inside}(e, x_3) \wedge \text{jar}(x_3) \wedge \text{loc-inside}(x_2, x_3)$
the boy looked in the jar at the frog inside it

$\exists e_1.\text{look}(e_1) \wedge \text{theme}(e_1, x_2) \wedge \text{frog}(x_2)$
the frog was looked at

...

These expressions all involve the boy looking at the frog, but there is a second set of possibilities where the frog is doing the looking:

$\exists e_2.\text{look}(e_2) \wedge \text{experiencer}(e_2, x_2) \wedge \text{frog}(x_2) \wedge \text{theme}(e_2, x_1) \wedge \text{boy}(x_1)$
the frog looked at the boy

$\exists e_2.\text{look}(e_2) \wedge \text{experiencer}(e_2, x_2) \wedge \text{frog}(x_2) \wedge \text{theme}(e_2, x_1) \wedge \text{boy}(x_1)$
$\wedge \text{loc-inside}(x_2, x_3) \wedge \text{jar}(x_3)$
the frog inside the jar looked at the boy

$\exists e_2.\text{look}(e_2) \wedge \text{theme}(e_2, x_1) \wedge \text{boy}(x_1)$
the boy was looked at

...

There are several others in each set, omitted for brevity, and there are still other possibilities involving both looking events at the same time (e.g., “the boy and the frog looked at each other”) or neither looking event (e.g., “there was a boy” or “the frog was inside the jar”).

A little syntactic knowledge plus even a single known word can quickly narrow the possibilities. Suppose the learner already knew the word for boy and that, in English, experiencers are likely to appear earlier in sentences than themes. He could then deduce that our example utterance (“the boy looked at the frog”) is more likely to have a meaning involving the boy doing the looking (our first set of candidates) rather than the frog, and other features such as shared attention and salience could further narrow the candidate set, permitting him to learn the correct mapping via cross-situational co-occurrence statistics.

Earlier computational models have dealt with this kind of referential ambiguity by exhaustively enumerating all meaning candidates and trying each one individually before eventually picking the candidate resulting in the most plausible meaning-utterance pair. The problem with this approach is that the number of possible meanings generally grows exponentially with the size of the scene description, quickly rendering exhaustive enumeration intractable. Just for our toy example presented here for illustrative purposes, there are well in excess of 20 possibilities, even after making several constraining assumptions. In fact, this toy example is already beyond what previously published work covers and it is dwarfed by the problem a model must face when dealing with the corpus of real data presented in Chapter 7, where there may be hundreds of meaning candidates per utterance.

To deal with this challenge, we take an alternative approach, working directly with the scene description rather than an exhaustive list of its various fragments, and rely on Dynamic Programming to cope with combinatorial overload. In some ways, the problem resembles that of finding the most probable parse in a conventional syntactic parsing setting. In the case of syntactic parsing, rather than explicitly enumerating all possible parses and comparing their probabilities, one typically relies on algorithms like CKY which exploit shared structure between different parses to turn this exponential problem into a polynomial one. The situation for word learning is not that different. Consider our example. Although there are dozens of possible meanings, most of them are quite similar with a great deal of structural overlap. For instance, in analyzing the

various ways in which the meaning representation

$$\exists e_1.\text{look}(e_1) \wedge \text{experiencer}(e_1, x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e_1, x_2) \wedge \text{frog}(x_2) \\ \wedge \text{loc-inside}(x_2, x_3) \wedge \text{jar}(x_3)$$

could possibly map to the words of "the boy looked at the frog", we would effectively perform all the work necessary for mapping the correct representation:

$$\exists e_1.\text{look}(e_1) \wedge \text{experiencer}(e_1, x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e_1, x_2) \wedge \text{frog}(x_2).$$

Thus, just as in the standard syntactic setting, it is possible to exploit structural similarities to avoid repeating identical computations, permitting a moderately powerful computational machine to tackle a much greater degree of referential ambiguity.

We accomplish this speed-up by employing a grammar-based model similar to the CCG approach of Kwiatkowski et al. (2012) and the PCFG model of Börschinger et al. (2011) but where the grammar is extended beyond the mapping between a sentence and its meaning to also model the compositional structure of the visual scene so that the entire problem is captured in a single grammar. Because sentence meanings and visual scenes have different properties from the ordered strings and trees for which traditional natural language processing formalisms have been developed, it is hard to directly extend PCFG-based models such as Börschinger et al. (2011). Furthermore, while CCG offers an elegant solution for mapping between meaning and sentence, it is much less clear how it applies to the mapping between scene and sentence meaning.

Thus, we turn to graph grammars, a PCFG-like formalism capable of generating unordered logical structures such as scene representations while simultaneously preserving many of the more convenient features of PCFGs. By representing meanings and non-linguistic contexts as unordered trees and modeling the mapping between them using graph grammars, it becomes possible to exploit these structural similarities between different candidate meanings using standard Dynamic Programming techniques familiar from string parsing. Furthermore, by using graph grammars which are capable of generating general graphs, not just the trees we will be using, we leave the door open for future extensions that may deal with more expressive meaning representation languages.

The product is not only a novel word learning model capable of testing the effect of syntactic bootstrapping in the face of greater ambiguity, but also a number of general algorithmic contributions such as an efficient parsing algorithm and a parameter estimation procedure that could find use in a wide range of applications. Finally, as an

additional contribution, we also present a newly annotated word learning corpus which allows us to quantify the amount of referential ambiguity in an easily measurable way and to test the performance of word learning models under different data and model assumptions.

Importantly, the model we present is less a proposal for how human learners might acquire word meanings and more of a demonstration of a framework under which we can test the effects of various assumptions and combinations of factors. Our model is designed to handle the computational overhead of a much greater level of ambiguity than generally assumed in the wider body of literature, requiring less reliance on salience and shared attention or other simplifying assumptions involving cognitive constraints or learning biases. However, this should not be construed as a cavalier rejection of these ideas and the supporting psychological studies. Rather, by refusing to build assumptions into the framework itself, the modeler himself is free to build whatever constraints into the model that best serve to explore his particular questions. Because the psychological assumptions are not built into the fundamental assumptions of the computational framework itself, they can be far more easily added or removed, permitting one to test the effects of different features or assumptions in isolation or in various combinations.

As an example, the model presented in Chapter 8 explores the power of the learning mechanism itself by testing its effectiveness with as few built in assumptions as possible, leading to a fairly high level of ambiguity. This allows us to explore the limits of the the notion of the poverty of the stimulus, i.e., the idea that data alone does not suffice and that language acquisition requires built in cognitive biases. In fact, the results demonstrate that, at least in theory, a learner could potentially acquire some semblance of a lexicon with far weaker built in constraints than previously tested in computational settings. The results, of course, say little about what human learners actually do or even whether they have the computational resources to mimic the simulation, and therefore do little to challenge the common view among the psychological community. However, the results suggest that perhaps if humans cannot learn as the model does, it is due not to poverty of the stimulus as much as to other constraints such as computational limitations or to differences in the scenario faced by the model versus that of children in the real world. The computational frameworks employed by previous models limited their abilities to explore this question, and thus we are able to speak more clearly to the power of statistical learning itself (as opposed to built-in constraints or notions of salience).

As for testing the effects of various features, this is largely left to future work, but as a demonstration, the model in Chapter 8 combines word learning with a crude level of syntactic learning, combined to demonstrate a synergistic effect that improves the quality of the lexicon in a manner consistent with hypotheses of syntactic bootstrapping.

This document is broken down into 9 chapters, covering the following topics:

Chapter 2 (Word Learning and Semantic Parsing Background) presents an overview of the main psycholinguistic results on word learning, and relates this to parallel developments in the semantic parsing literature. This chapter is evenly split between, in the first half, the main ideas circulating in psycholinguistics, and in the latter half, an overview of work in semantic parsing. Interestingly, while the objective of semantic parsing work has been somewhat different, an increasing interest in reducing reliance on annotation, coupled with an increasing focus on statistical learning in psycholinguistics, have led to a certain convergence of ideas. In fact, some researchers have even proposed semantic parsing as an alternative view of word learning. This chapter attempts to highlight these points of convergence as it conducts a somewhat broader survey of the psychological and semantic parsing literature in general. We also present a case study of a competing semantic parsing-inspired word learning model to highlight the source of the computational complexity our approach is designed to address.

Chapter 7 (Frog Stories Corpus) describes a new set of annotations for a multilingual psycholinguistic corpus and follows up with a corpus analysis quantifying the degree of potential ambiguity that human word learners face. Due to divergent interests in semantic parsing and psycholinguistics, prior to this work, it was difficult to identify a corpus suitable for our purposes of exploring the syntax in resolving referential ambiguity in word learning. Thus, we turn to hand annotating an existing psycholinguistic corpus known as the Frog Stories (Berman and Slobin, 1994). These annotations not only facilitate new computational experiments but also have the added benefit of permitting us to quantify in a concrete way the amount of referential ambiguity that a human learner may face. We discuss in detail the many of the assumptions and cues exploited by our model in the experiments in Chapter 8 and quantify their impact on the raw number of meaning candidates. In summary, this chapter describes the annotation scheme chosen, provides a brief justification for the annotation choices

made, and presents the corpus analysis quantifying the effects of various constraints on reducing referential ambiguity.

Chapter 3 (Grammar Background) lays out the necessary definitions and basic theoretical properties of the context-free synchronous grammars we use. Our grammars are close enough cousins to PCFGs that the reader is likely to already be familiar with many of the algorithmic foundations, such as the inside-outside algorithm, but there are a number of new notational conventions required for precisely describing how these technologies apply in this new setting. Additionally there are some extensions to synchronous and probabilistic grammars that offer increased modeling flexibility which we make use of in Chapters 6 and 8. One innovation in particular includes a generalization of probabilistic grammars to allow rule probabilities to be modeled by entire Bayesian networks, allowing for the modeling of rule-internal independence assumptions which can alleviate many sparsity problems commonly encountered in synchronous grammars.

Chapter 4 (Parsing Unordered Trees) follows up with a description of a novel algorithm for parsing ranked unordered trees with a particular sub-class of graph grammar. There are a few papers describing general graph parsing algorithms and their complexities. However, the general case is far more powerful than our relatively modest needs, and, in the most general case, leads to an exponential time algorithm. Instead, we outline a procedure specialized for the ranked unordered trees necessary for parsing the Frog Stories corpus that remains fairly general while permitting a much more efficient solution. In fact, the algorithm includes a number of optimizations that are applicable to more general graph parsers, potentially leading to major performance improvements that could make the difference between a theoretically interesting formalism and a practical tool for natural language parsing.

Chapter 5 (Inference in Multi-weighted Grammars) describes a variational algorithm for applying Bayesian inference to the multi-weighted grammars introduced in Chapter 3. Just as the Bayes net grammars are a generalization of standard probabilistic context-free grammar, the inference algorithm includes the standard mean-field variational Bayesian inference approach as a special case.

Chapter 6 (Semantic Parsing) presents a model based on the grammatical

framework described in Chapter 3 to the traditional problem of semantic parsing. The problem is simpler than that of word learning, since the learner is presented with an observed meaning representation for each sentence, something that a child must infer from context as he learns word meanings. However, it can be interpreted as a sub-problem of the word learning task, setting the stage for the full word learning model presented in Chapter 8. At the same time, we demonstrate the effectiveness of expressing semantic parsing in a standard grammar framework, showing performance improvements over several problem-specific state-of-the-art solutions.

Chapter 8 (Word Learning) incorporates the semantic parsing model of Chapter 6 using the grammar framework of Chapter 3 into a general word learning model, applying it to the Frog Stories corpus. In the process, we demonstrate that roughly the same model works just as well for semantic parsing as for previous word learning tasks (without syntactic bootstrapping). Finally, turning to the problem of word learning with syntactic bootstrapping, we present experiments testing the impact of syntactic cues under highly ambiguous settings, and measure the effectiveness of joint learning on resolving this referential ambiguity in combination with several other sources of information and constraining assumptions.

Chapter 9 (Conclusion) concludes with a discussion of some additional implications of our results, highlights several additional avenues for future inquiry, and suggests other applications of the grammar framework.

Chapter 2

Word Learning and Semantic Parsing Background

Given the apparent difficulty of the various entangled problems involved in learning the meanings of words, it is amazing how regularly children succeed. By one year of age children are typically already producing words, likely understand far more than they produce, and are rapidly adding to their repertoire. What is the nature of the mechanism they use to accomplish this feat? Perhaps they rely on innate language-specific learning biases, or maybe there are clues hidden in the data that simplify the task, but to what extent might these factors ease the learning problem? In other words, how powerful is the learning mechanism itself? Many psychologists have spent their careers searching for answers to these questions, but the last question has begun to attract greater attention with a relatively recent resurgence in interest in theories relying on statistical learning.

While psychologists explore the nature of the human faculties required for solving the problem, researchers in natural language processing (NLP) have pursued similar questions but focused more directly on the learning problem itself and on the computational techniques required for its solution. From the inception of the field, NLP has sought to automate language understanding, enabling computers to extract and act directly on linguistic information. In a task known as "semantic parsing", computer systems attempt to translate sentences in natural languages like English into unambiguous formal languages that are more readily interpretable by machine. Given the magnitude of the task of manually engineering a broad coverage grammar and lexicon from scratch, work typically relies on machine learning to automatically construct this word-to-meaning map directly from data, much as children seem to rely on their own

learning mechanisms. Of course, the data fed to the mechanical learner differs significantly from the world the child experiences, but an increasing interest in larger, richer data sets, with simultaneously less reliance on hand annotation, has narrowed this difference somewhat. Thus, while psychologists have begun to explore how the existence of a powerful statistical learning mechanism might impact word learning, semantic parsing experts have converged on a very similar problem as they have gravitated toward semi- or unsupervised learning with richer input that, in some cases, resembles more closely what children experience.

Given the similar focus and, in some cases, computation techniques employed by researchers in the two sub-areas of semantic parsing and word learning, the situation seems ripe for cross-pollination. Are there ideas from psychology that can help to better define the semantic parsing problem, and are there computational techniques that can shed light on the situation of the human learner? This chapter seeks to shed some light on this question, examining the previous work in both fields for promising points of intersection.

2.1 Word learning

In analyzing the situation facing children, psychologists have identified several layers to the problem, cross-cutting all areas of linguistics and extending into other more general aspects of cognition such as perception, category learning, and theory of mind. Children must learn about the surface forms of the language, from fairly basic things like categorizing speech sounds into the phonetic categories of the target language, to identifying words in an unbroken stream of speech, learning how individual words are constructed from smaller morphemes, and how such words combine to form phrases. However, the term “word learning” is usually used to refer primarily to the problem of learning to associate these surface forms with meanings.

Somehow, a child must learn that the word “doggy” refers to the stuffed animal in mom’s hand, as opposed to the many other things that are present such as the chair, the table, a shoe and so on. This is the problem of referential ambiguity, where the listener must resolve which specific object is being talked about. In fact, the child must rule out other possible candidate referents such as *leg* or *tail*, two other things that are both present when the child hears the word “doggy”. Assuming the child has somehow identified both the surface form and the referent, however, there is still much more to a word meaning than a surface form-object pair. For instance, the child

must learn that “doggy” is more than a label for that specific stuffed animal; “doggy” can be used for a whole category of things. The child must somehow infer that the category includes actual animals like Terriers and Dalmatians but not other four legged mammals like cats. There is a whole hierarchy of categories that include that particular stuffed animal, and somehow the child must determine the correct level of abstraction (i.e., that “doggy” refers to canines rather than mammals in general). Actually, there are other hierarchies one might consider, given that the dog stuffed animal is a kind of doll which is in turn a toy and so on, and the child must resolve the correct level of abstraction within each hierarchy. Furthermore, different words can be used to refer to construct the same mental scene, where the only difference is the emphasis that the speaker places on certain entities. For instance, the sentence “the boy received the jar from his father” can describe exactly the same set of situations as “his father gave the jar to the boy.” It seems that learning the distinction between the meanings of the verbs “receive” and “give” must somehow involve reading the speaker’s mind. Thus, there is a tremendous amount of ambiguity, both referential and categorial, that a child must somehow resolve, a seemingly intractable task which has led psychologists to propose a number of innate constraints and heuristics to guide the word learning process.

2.1.1 Learning biases and constraints

One proposal that could help resolve some of the ambiguity is an assumption that different word forms tend to be used to refer to talk about different concepts, a principle known as mutual exclusivity (Markman and Wachtel, 1988; Littschwager and Markman, 1994). If the learner encountered a new word and narrowed down the possible referents to two objects, a familiar one for which a word was already known and the other completely novel, mutual exclusivity would match the novel object with the novel word. While such a heuristic fails in the case of homonymy, where the same word form has two different meanings, or synonymy, where different words have the same meaning, such cases seem to be fairly rare, so one would expect the heuristic to be helpful in general. Of course there are other phenomena that mutual exclusivity cannot explain on its own.

As an example, the part-of relations alluded to above where “dog” and “tail” both refer to the same object, albeit different subsets of the object, so mutual exclusivity fails even though “dog” and “tail” are neither homonyms nor synonyms, and part-of relations are frequent enough that they are harder to dismiss. This difficulty has led

to the proposal that children may also make use of a whole-object bias, according to which they would assume by default that “dog” refers to the whole stuffed animal rather than to its tail or leg (Carey, 1978; Mervis, 1987). This whole-object bias seems to help with deciphering child-directed speech.

Similarly, additional biases have been proposed for resolving the appropriate level of abstraction. While mutual exclusivity and the whole-object bias offer little help, additional biases could help resolve whether “dog” is simply a label for the particular toy, for toys in general, or the *animal* concept. Markman (1989) argues that children could rely on a “taxonomic bias” to map words by default to concepts that optimize utility with respect to a certain level in the hierarchy. Such high utility concepts are known as basic level categories in the psychological literature (Rosch et al., 1976). A taxonomic bias would help rule out learning labels for conceptually strange categories such as “all spotted animals, but not leopards, plus chairs” since it is hard to imagine such a category being very useful. Additionally, it should also bias learners away from other plausible categories which are less frequently used. One might assume that “dog”, for example, refers to canines in general, and is not just a name for that particular toy, or Terriers only, or all mammals, because *canines* optimizes some metric of utility that the others do not.

Related to the taxonomic bias, researchers have also observed that children tend to generalize words to novel objects of the same shape, but not necessarily to objects with other shared features such as texture, color, or material (Markman and Wachtel, 1988; Landau et al., 1998; Smith et al., 2002). This bias could help select between different overlapping hierarchies. Thus, rather than generalizing the label “dog” to other objects with the same color (spotted animals), a learner would tend to generalize to other four legged animals with a tail.

2.1.2 Helpful features

At the same time as proposing biases that could help constrain the learning problem to some more tractable set of hypotheses, psychologists have also explored questions of exactly what information is available to a child. Perhaps there is some cue or combination of cues in the data that could successfully guide a learner even in a vast space of mostly incorrect guesses. Sensitivity to such features could either complement innate learning biases or allow the learner to avoid the use of certain constraints altogether.

Whether it is because (1) higher frequency simply means the learner has more

learning opportunities or because (2) humans make explicit use of frequency numbers, word frequencies are one type of correlate with successful word learning. For instance, arguing for the first case, Huttenlocher et al. (1991) observed that learners who receive greater exposure to speech in general tend to acquire a larger vocabulary. Similarly, it has been observed that nouns tend to be learned far more quickly in English and other languages where nouns are more frequently occurring in prominent locations such as the end of the sentence (Gentner, 1982). However, in some other languages where verbs are more frequent such as Mandarin (Tardif et al., 1997), Korean (Choi and Gopnik, 1995), or Tzeltal (Brown, 1998), children learn verbs far earlier than they tend to do in English. On the other hand, it is possible that frequency could have a more subtle impact on learning, possibly providing an alternative account for the shape bias, relying on learning instead of assuming an innate constraint. For example, Smith et al. (2002) exposed subjects to words for categories of objects with a common texture and found that this increased the likelihood that they would generalize new words across texture-based categories, counter to what one would expect under a shape bias.

Frequencies could also be exploited in cross-situational learning (Pinker, 1989; Gleitman, 1990; Yu and Smith, 2007; Akhtar and Montague, 1999), the idea that children may be able to exploit consistencies in word-object co-occurrences across multiple scenes to help eliminate ambiguities. That is, if a child observed that mom tended to be holding the stuffed dog at the same time as saying “doggy” but was far less likely to say “doggy” when holding some other toy, say a ball, it would seem reasonable to prefer the mapping that associates “doggy” with the stuffed animal rather than the ball. As an example, (Smith et al., 2011) exposed subjects to a sequence of scenes with up to nine potential referents simultaneously with novel isolated words. At test time, subjects were shown the full set of objects for which words were given during training and asked to point out the best object for each new word and found that learners performed fairly well, well above chance in spite of the fact that the experimental setup seemed to deprive learners of other information that might allow them to use alternative strategies such as mutual exclusivity.

It has also been argued that word learning is fundamentally a social phenomenon, and thus social cues such as body language should play a crucial role. As an example, it seems that children are sensitive to and capable of following the direction of a speaker’s gaze to infer which object is being discussed (Baldwin, 1993; Baldwin et al., 1996). This observation is closely associated with the idea that children are able from a relatively young age of inferring and reasoning about others’ mental states, allowing

them in turn to infer the intended referents for words (Tomasello, 2001; Bloom, 2000).

Salience is another useful feature type. For instance, learners seem to learn words faster if they appear in prominent locations such as at the end of sentences. This is an example of accidental salience, where it happens to be a feature of the language; hence in some languages children learn verbs earlier than others possibly because of the accident of word order (Tardif et al., 1997; Choi and Gopnik, 1995; Brown, 1998; Gentner, 1982). In other cases caregivers may deliberately exploit salience to facilitate the child's understanding and learning. For example, parents may make a special effort to engage very young children by talking about whatever the child shows greatest interest in, so that the child's gaze dictates the topic of conversation: the most salient object in the child's field of vision. Infant directed speech has many features that may relate to salience: exaggerated pauses, vowels, slow speech rate, and pitch contours. At the very least, such exaggerated speech does seem to attract children's attention, since children show a preference for listening to it (Fernald, 1985; Cooper and Aslin, 1990), and it also seems to facilitate word learning (Graf Estes and Hurley, 2013). Infant-directed speech may also serve to attract the child's attention to increase the salience of particular words. Fernal and Mazzie (1991) observed that new words in particular are more likely to occur at such positions of prominence, possibly helping the child identify the novel word to be identified with the new topic of conversation. Another effect of this exaggerated prosody is that it can help to contrast function words vs. content words, since content words are more likely to be stressed in this way.

Finally, syntax also provides a set of features that can help in learning new words, a phenomenon that has been called syntactic bootstrapping (Gleitman, 1990). Naigles (1990) demonstrated that 2-year-olds were able to identify transitive verbs with scenes depicting causative relations, e.g., "the rabbit is gorping the duck," as opposed to non-causative relations like "the rabbit and duck are gorping," arguing that children exploit syntactic information (verb arity) to correctly match transitive verbs to their referents. Similarly, Gillette et al. (1999) conducted an experiment also showing that knowledge of nouns can help learn verb meanings. Lany and Saffran (2010) further demonstrated syntax can help learn nouns, not just verbs, exploiting a variety of case marking to help identify nouns with their referents. In their experiment, subjects were exposed to two different categories of words in an artificial language, one used for describing vehicles, another for animals, where each type of word was preceded by a particular word that correlated with its category: vehicles got one prefix, animals another. Subjects showed greater success at learning words in this configuration than when the prefixes were

random. By exploiting such syntactic features, it is easy to see how learning could accelerate as vocabulary grows, helping to quickly rule out a great deal of ambiguity that would otherwise interfere.

2.1.3 Computational models

There are a number of computational models of word learning in the literature, but because the problem is so cross cutting, touching on virtually all aspects of cognition, models necessarily focus on different aspect of the problem.

There are models that assume the input consists entirely of isolated words, and some, in fact, that train on observed word-referent pairs, removing all referential ambiguity. These models instead tend to focus on category learning. For instance, there are the connectionist models of (Regier, 2005; Plunkett et al., 1992; Schafer and Mareschal, 2001; Colunga and Smith, 2005; Gasser and Smith, 1998; Smith, 2000a). These models simultaneously learn sound categories, mapping different instances of the same word to the same form, while also learning to identify the relevant features of the referent to identify with the meaning of the word. Regier (2005), for instance, exhibits behavior consistent with the shape bias. Li et al. (2004; 2007) presents an incremental, associative model, demonstrating a vocabulary spurt-like pattern, an often observed acceleration in rate of acquisition the occurs after the learner surpasses some threshold. Another model demonstrates fast mapping, another phenomenon observed in children where they are sometimes able to learn a new word after a single exposure Horst et al. (2006).

Aside from the connectionist models, there are also a few others that assume the input consists of isolated word-object pairs. The “competition-based models” which formulate categories by selecting the relevant features by contrasting training examples (MacWhinney, 1989; Merriman, 1999). These models naturally exhibit behavior consistent with mutual exclusivity. Other models rely on similarity metrics, identifying different instances of the same word, or generalizing that word to different objects if the word instances and objects are similar enough (Landau et al., 1988; Roy and Pentland, 2004). There is also the Bayesian model of Xu and Tenenbaum (2007), which relies on the frequency with which certain object features are associated with a given word meaning to generalize across different words and objects. By this means, the model is able identify the correct level of abstraction and to select between different overlapping category hierarchies, and is capable of learning things like the shape bias

directly from data and word frequencies.

Other models deal with referential ambiguity, working with words in context (i.e., whole sentences rather than isolated words) coupled with multiple candidate referents and relying on cross-situational information to resolve ambiguity. Siskind (1996) represents one of the earliest examples of these models, which learns by identifying for each word a minimal set of features that is consistent with the different situations, where a "situation" is a pairing between a scene and a sentence. However, as a rule-based model, it was quite vulnerable to noise. Other models dealing with referential ambiguity usually rely on symbolic meaning representations instead of raw sensory data or the feature vector representations used by the category learning models. Fleischman and Roy (2005) dealt with the part-of relationship for identifying individual steps of a plan with the meaning of a specific sentence, learning, for example, that "find axe" maps to a sub-problem within a broader planned action like the one identified with "get axe" (one must locate the axe before acquiring it).

Others tend to frame the problem in more or less the same way where the input typically consists of transcripts of the utterances coupled with the set of objects present at the time.

(2.1) what does the doggy say ?
{ ball, dog, pig, mirror }

Lexical entries in these models consist of word-object pairs, where the object serves as the meaning of the word. The model of Yu (2005) treated this as a translation problem, applying IBM model 1 (Brown et al., 1993) to learn word-object association probabilities and applying thresholds. Fazly et al. (2010) presents another model, which learn incrementally and demonstrates mutual exclusivity, fast mapping, and a vocabulary spurt. Frank et al. (2009) introduced a Bayesian model with an explicit representation of the lexicon as a set of word-object pairs, which could integrate social cues into the learning process. By placing a prior over the size of the lexicon, biasing in favor of smaller lexicons, this model showed that behavior such as mutual exclusivity and fast mapping could be seen as a result of a preference for a sparse lexicon.

Other Bayesian models followed. Jones et al. (2010) translated the model of Frank et al. (2009) into a product of Dirichlet-multinomials, allowing them to integrate word learning with the word segmentation model of Goldwater et al. (2009). This, in turn, was translated into a PCFG framework, allowing for greater flexibility in exploring other varieties of learning synergies in the form of phonological learning and word

structure (Johnson et al., 2010), and later the model was extended to include social cues (Johnson et al., 2012), and discourse structure (Luong et al., 2013).

There have also been some models that attempt to incorporate syntactic knowledge into cross-situational word learning. Assuming that a learner already acquired certain basic syntactic word categories, Alishahi and Fazly (2010) demonstrated that such knowledge helps, since only certain categories are likely to map to certain kinds of semantic concepts (e.g., nouns with objects). By assuming fairly sophisticated knowledge about the syntax-semantics interface, Niyogi (2002) showed how these rules about how word meanings combine within a sentence to constrain the meanings of unknown words. However, it is unclear how such a model would operate given only incomplete or noisy syntactic knowledge, or how the learner might acquire such a sophisticated syntactic model, and it was only demonstrated for a small set of hand picked examples. Other models jointly learn syntax with meanings, such as Yu (2006), which simultaneously categorizes words by common sentential contexts and maps words from these categories to objects such that words from a common category are more likely to be associated with objects of the same type. Maurits et al. (2009) also jointly learns word order and meanings, but assumes a fairly constrained set of possible word orders.

As apparent from the preceding discussion, there are numerous models of how cross-situational learning can be combined with other features or learning biases to resolve referential ambiguity, but thus far the amount of ambiguity considered has been fairly limited. Typically, there are no more than a handful of candidate objects, something often necessitated by the inherent computational complexity underlying the assumptions of the models on sentence meanings. In particular, these models essentially explore all possible subsets of objects, resulting in a set of candidate meanings that is exponential in the size of the scene. Thus, hampered by this computational constraint, there has been little work exploring the limits of cross-situational learning. How much ambiguity is too much?

2.2 Semantic parsing

Largely independent of the psychological community, researchers in natural language processing have been pursuing solutions to a related problem, automating the extraction of “meaning” from natural language by computer, a task commonly referred to as *semantic parsing*. Similar to the computational models of word learning, these

systems automatically learn associations between words and symbolic meaning representations. Semantic parsing as a field has been understood to cover many different tasks, where the unifying objective is that of mapping a sentence to a representation of its meaning expressed in a machine-interpretable formal language. For our purposes, however, we use a more narrow definition centered on a single basic task where the typical input to the learning algorithm consists of a set of sentences paired with their meaning representations such as the following:

(2.2) How many states does the Colorado River flow through?
answer(count(state(traverse(river(riverid(colorado))))))

This task is sometimes referred to as *supervised semantic parsing*, since the system is given an explicit meaning representation for each sentence at training time, to contrast with other training schemes some of which are discussed in the latter portion of this section. By learning a correspondence between individual segments of the meaning representation with the words of the accompanying sentence, systems can generalize to novel sentences, permitting the computer to act directly on natural language to answer users' questions just as it might execute a database query expressed in a computer language like SQL, for example.

Many of the problems that a child faces in word learning are present in this semantic parsing scenario as well. In particular, given no other information other than sentence and meaning expression, there is a fair amount of referential ambiguity to resolve. A model must learn that "how many" signifies a question, and that the answer is expected to be a number, signaling a call to function *answer(NUM)*, that "many states" indicates that the answer requires counting states, *count(STATE)*, that the "Colorado River" signifies *riverid(colorado)*, and that "flow through" indicates a call to the *traverse(RIVER)* function. Without any additional information, any word in the sentence could map to any symbol in the meaning representation. In fact, the meaning-word map could even be many-to-many with a single word relating to any number of symbols in the meaning representation (or none) or where a single symbol is expressed by zero or more words, leading to an exponential number of potential mappings, most of which are not very useful. The model is ideally expected to rule out the poor mappings purely by observing consistencies across multiple training pairs in exactly the same way that a child might exploit cross-situational information to learn word meanings.

Semantic parsers quickly rule out a large number of candidate mappings by making an assumption that amounts to something known in the psychological literature

as semantic bootstrapping (Grimshaw, 1981; Pinker, 1989). In semantic bootstrapping, semantic information is used to assist in the inference of syntactic structure. For instance, in a sentence like “the dog broke the jar”, knowing that the jar is the semantic argument of the *break* predicate might suggest that it should also appear as a syntactic argument of the verb “broke”. In the case of semantic parsing, systems effectively assume a structural relationship between the meaning representation and the syntactic analysis of the sentence, and then perform a sort of grammar induction, relying on the structure inherent in the meaning representation for guidance. Because *riverid(colorado)* is an argument of *traverse(RIVER)* in the meaning representation, a system may also tend toward derivations of the sentence that generate “Colorado River” as a dependent of “flow”. In this way, rather than inferring a grammar based on the words alone, semantic parsers infer a grammar over meaning-sentence pairs, a somewhat different objective than in purely unsupervised syntactic parsing. Similarly, some semantic parsers may also exploit regularities in the learned syntax-semantics map to generalize across predicates which can in turn allow syntactic knowledge to assist in word learning (i.e., syntactic bootstrapping).

2.2.1 Two general approaches

Most semantic parsing models can be roughly categorized under one of two different approaches. One approach, which we will refer to as the tree transformation approach, assumes that meaning representations are either themselves trees or that they can be parsed by an unambiguous grammar to identify a tree, and that this tree closely resembles, and can be used to produce, a parse tree for the sentence (Jones et al., 2012b; Lu et al., 2008; Kate and Mooney, 2006; Wong and Mooney, 2006; 2007; Tang and Mooney, 2000; Jones et al., 2012a). In this case, the meaning representation (or its parse), is a tree where each node is typically identified with exactly one function symbol in the meaning expression. The strategy is to transform this meaning representation tree into a derivation for the sentence by first reordering it and then attaching words as needed to produce a projective parse tree. Conceptually, the idea is related to the syntactic theory of Transformational Grammar (Chomsky, 1957; Cooper, 1975), where an underlying form (i.e., the meaning) is transformed to produce the surface form. In principle any transformation is permissible, but to keep things tractable, permutations are usually restricted to local rotations among sibling nodes. Where words attach is dictated by maintaining some kind of meaning-word map that specifies which mean-

ing symbols correspond to which words: attachment is permitted between a given node in the meaning representation tree and a word if and only if the corresponding meaning symbol and word pair are in the map. Assuming that the resultant parse tree for the sentence is projective significantly cuts down on the number of possible meaning-word links that might be proposed, reducing the search space and making the problem more tractable. The transformation from meaning representation to sentence is governed by a formalism that can be interpreted either in terms of synchronous context-free grammar or tree transducers. The systems primarily differ in the details of how the meaning-word map is structured and in the learning mechanism employed.

The other main approach employs Combinatorial Categorial Grammar (CCG) (Steedman, 2000) to identify a homomorphism between meaning representation and natural language syntax (Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2010). Under this approach, CCG lexical items are inferred by decomposing the meaning representation into several smaller lambda expressions and mapping words to those sub-expressions while simultaneously assigning a syntactic category to the pair. The resultant CCG rules combine in ways dictated by the syntactic categories to simultaneously derive the meaning representation and the sentence. As per the CCG theory, the lexicon in this case consists of these labeled meaning-word pairs. There is an exponential number of ways of decomposing a given meaning representation, and training consists of identifying a small subset of highly reusable rules to enter into the lexicon. To make things tractable, systems typically restrict search to a certain subsets of meaning representation decompositions, greatly reducing the search space.

Although the two approaches may appear very different on the surface, the basic underlying idea is similar. They look different because while the transformation-based approach deals in meaning representation trees, the CCG-based approach deals in decompositions. However, this difference disappears when one realizes that a parse tree is simply another way of representing a particular decomposition.Thus, the key difference between the two is really just that the transformation-based approach starts with a single decomposition for the meaning representation and then transforms it to produce a set of different decompositions, while the CCG-based approach starts off by enumerating the set of decompositions directly. In practice, however, CCG-based systems usually apply fewer restrictions to the decompositions considered than tree transformation-based systems do to the final transformed trees. As a result, transformation-based systems are often more efficient in terms of computing resources while CCG-based systems explore a larger space of possible grammars, potentially producing better results.

This observation is born out empirically by the experiments reported in Chapter 8, where the model of Kwiatkowski et al. (2010) performs quite well but takes around 8 hours to run to completion on a single language of the data set, while the systems of Wong and Mooney (2006) and Lu et al. (2008) perform somewhat less well but complete in under an hour.

Both approaches are capable of simulating syntactic bootstrapping effects. For instance, in the transformation-based approach, different weights can be assigned to different permutations whereby models can learn basic facts about word order in a given language. Similarly, the syntactic labels of the lexical items in the CCG approach specify the order in which words combine to form a sentence (word order, in other words). If general patterns are learned which generalize to novel verbs, models under either approach can learn the canonical ordering among the verb and its subject and objects, potentially facilitating fast mapping and generally guiding convergence to more accurate lexicons.

Finally, while these models can easily be interpreted as computational word learners, they operate under much less ambiguity than children often face.Specifically, these models assume training items are sentence-meaning pairs, where the model is given the true meaning of the sentence and the referential ambiguity consists entirely of determining which words go with which parts of the meaning representation. However, this seems like an oversimplification when one considers the kind of information a child receives. For one thing, it is difficult to imagine that a child always knows exactly what a sentence means. If a child's caregiver were reading a story, for instance, the child has an entire picture where a large number of things might be described. The caregiver may point to the character being talked about, but is this character the agent or the patient or something entirely different, and which event is being described? Thus, a more realistic model should ideally be capable of dealing with a greater degree of ambiguity. The next section discusses more recent attempts to address this sort of problem which build on but go beyond our restricted definition of semantic parsing.

2.2.2 Alternative sources of supervision

In the scenario outlined so far, the model receives two pieces of information for each training item: the words of the sentence and the sentence's meaning representation. However, many have also experimented with other forms of supervision. For instance, one might expect that the gold syntactic analysis of a sentence could be leveraged

to improve the quality of grammars that jointly generate the meaning-sentence pair. In fact, several such systems have been designed (Ge and Mooney, 2005; Le and Zuidema, 2012; Jones et al., 2012a). Often times, these approaches resemble the inverse of the transformation-based approach, starting from the syntactic parse tree and generalizing it to cover the meaning representation. However, viewed as a model of word learning, this approach effectively assumes a learner who already possesses a mature grasp of syntax, but while there is evidence to suggest children begin learning things like basic word classes fairly early in life, they certainly do not master syntax at the level assumed by these models until much later.

At the same time, driven by the goal of scaling up to larger and more diverse domains, work has also focused on reducing dependence on meaning representation annotations, which typically require expensive manual labor. Most methods seek to exploit task-specific features of the target application of the system. For instance, for systems that seek to automatically learn natural language interfaces to databases, the learning algorithm can make use of the database itself instead of explicit meaning representations (i.e., queries in a database language like SQL) by testing guessed meanings to see whether the inferred query returns the expected answer (Clarke et al., 2010; Liang et al., 2011). In this case, sentences are effectively paired with the gold query *result* rather than the *query itself*. Other systems seek to learn to map natural language instructions to the actions they describe (Branavan et al., 2010; 2011). Such systems exploit information about success at performing the desired task as a measure of accuracy and can optimize this metric. Another line of work focuses on mapping navigational instructions to movements within a simulated world (Vogel and Jurafsky, 2010; Chen and Mooney, 2011; Kim and Mooney, 2012; 2013). Again, systems can simulate following the instructions and use success or failure (i.e, arrival at the expected destination) to inform learning. Alternatively, given current location and target destination, systems can plot out a path between the two and attempt to align individual steps to segments of the natural language instructions. Thus, instead of meaning-sentence pairs, the learner effectively receives sets of candidate meanings for each sentence and must simultaneously infer the correct meaning for each sentence and the meanings for each individual word in the sentence.

In fact, this last approach of simultaneously inferring the best meaning candidate from a set and the map from meaning to sentence is a very general strategy that can in theory be applied to any domain. It has been applied to a sportscasting domain, where the model attempts learn to identify natural language descriptions of how a

ball is passed around a field in a simulation to the underlying state information of said simulation (Chen et al., 2010; Kim and Mooney, 2010; Chen and Mooney, 2008; Börschinger et al., 2011). It has also been applied to a navigational domain (Artzi and Zettlemoyer, 2013) and child-directed speech (Kwiatkowski et al., 2012). In most cases, the model was adapted from a previous semantic parser designed to train from observed meaning-sentence pairs where there is no ambiguity about what each sentence means. Systems can do this in either a pipelined fashion where first the meaning is selected and then it is mapped to the sentence, or by jointly inferring the candidate meaning and the mapping. Pipelined systems first employ a separate alignment model to attempt to identify the true meaning representation among the set of candidates. This alignment model may itself be based on a semantic parser or could be a different model specialized for the task such as Liang et al. (2009). Following the alignment step, the semantic parser is then trained as before using these guesses. In the case of joint inference, the semantic parser must devise a map from each possible meaning candidate, effectively parsing the sentence repeatedly, once per meaning candidate. This need to parse every candidate can be more expensive than the pipelined approach, but it is often only slightly so since even in that setting each meaning candidate must be analyzed separately before a guess can be made. However, the pipelined approach potentially introduces noise by forcing a hard choice even when there is a great deal of uncertainty that the joint learner could in theory account for by hedging its bets with multiple possible choices, each weighted according to some confidence score.

Any system that must separately analyze each meaning candidate is bound to scale poorly with increasing uncertainty (i.e., as meaning candidates increase), and semantic parsing is typically fairly expensive even when the gold meaning representations are observed, so most systems limit the set of meaning candidates to just a handful. This is one area that the model proposed in Chapter 8 is intended to address, allowing us to better explore word learning performance under much higher levels of uncertainty.

Some work has even been done on learning without meaning representation annotations by exploiting domain knowledge (Goldwasser et al., 2011; Poon, 2013; Reddy et al., 2014). The domain knowledge in question would typically not be available to child language learners, however. For one thing, these approaches often rely on string similarity between the words of the natural language and the symbols of the target meaning representation language (i.e., words such as “populous” or “populations” are biased to map to the symbol *population* in the meaning representation language). In addition to string similarities, Goldwasser et al. (2011) employs various heuristic met-

rics again not necessarily well suited for simulating the child's situation to gauge parse quality at each iteration of a self-training procedure. Poon (2013) and Reddy et al. (2014), on the other hand, make use of supervised syntactic parsers pre-trained on outside corpora and rely on the parse structure to map syntactic relations to database relations. These approaches show that cross-situational learning is not strictly necessary for learning a word-meaning map, but they rely crucially on information unavailable to child language learners, making poor candidates as cognitive models of word learning.

As methods develop for reducing annotation overhead, researchers have also begun working on large scale knowledge bases such as Freebase, which contains over 39 million topics Doe (2014). Because annotating the vast amount of web-scale data required for training would be prohibitive, systems typically either automatically extract meaning representations Cai and Yates (2013) or employ similar techniques to those already described to reduce dependency on such information (Krishnamurthy and Mitchell, 2012; Kwiatkowski et al., 2013; Reddy et al., 2014), or some combination of the two (Berant et al., 2013). Furthermore, just as with the task-specific approaches, they also tend to exploit rich syntactic information or string similarities between words and the symbols of the knowledge base, or assume all sentences are knowledge base queries. Thus, although results have been impressive, these models are also inappropriate as cognitive models.

Finally, there are still other lines of work that seek to ground words in raw sensory data (Krishnamurthy and Kollar, 2013; Kollar et al., 2010; Matuszek et al., 2012). These techniques require simultaneously solving various perceptual problems such as object recognition in raw visual data. Research in these areas have whole fields dedicated to solving the various related problems which add noise to and often dominate the word learning problem. In order to focus on the word learning task, we will set aside sensory data for our work here and instead map to symbolic data as described in Chapter 7.

2.3 Case study: CCG

Kwiatkowski et al. (2012) propose a word learning model based on Combinatory Categorial Grammar (CCG) to explore synergies in word learning and syntactic grammar induction. The model attempts to tackle the problem where there are multiple candidate meanings for any given sentence and both the sentence meaning and the individual word meanings must be simultaneously inferred. As a grammar-based model that

jointly infers sentence and word meanings, the artificial word learner they describe is similar in spirit to the model we propose in Chapter 8. At the same time, being based on CCG as opposed to our synchronous grammar approach, their model also represents a competitive example of the other leading approach to semantic parsing. Both the similarities and differences make it an interesting candidate for a more in depth look. We pay particular attention to computational complexity, highlighting the source of the complexity, since this effectively bounds the amount of ambiguity a model can tackle, and ultimately we wish to reduce this computational overhead to make way for learning performance itself as the main focus of study.

Performing grammatical inference using CCG, a strongly lexicalized formalism, results in a grammar that doubles as a syntax-enriched lexicon, where each lexical entry consists of three parts, a word, its syntactic category that dictates how words combine into phrases, and its meaning in the form of a logical expression, represented in typical CCG notation as

$$\text{word} \vdash \text{syntactic category} : \text{meaning representation}.$$

Some example entries might include

$$\begin{aligned} \text{the} &\vdash \text{NP/N} : \lambda f \lambda x.\text{the}(x, f(x)), \\ \text{dog} &\vdash \text{N} : \lambda x.\text{dog}(x), \\ \text{licked} &\vdash \text{S}\backslash\text{NP/NP} : \lambda x \lambda y.\text{lick}(y,x), \text{ and} \\ \text{girl} &\vdash \text{N} : \lambda x.\text{girl}(x), \end{aligned}$$

where the syntactic component of each entry specifies (1) the types of words with which it can combine, (2) the relative position of those words, and (3) the resultant type of the combination. For instance, type NP/N indicates that the determiner "the" combines with a noun (type N) such as "dog" to the right to produce a noun phrase (NP). Precisely how lexical entries combine to form phrases are dictated by combinators, of which there are two types in their model: application, and composition.

Application, the simpler of the two combinator types, takes its name from the fact that it involves a applying the lambda expression that represents the meaning of one word to the expression associated with the other.

$$\text{X/Y} : f \quad \text{Y} : g \implies \text{X} : f(g) \qquad (>)$$

$$\text{Y} : g \quad \text{X}\backslash\text{Y} : f \implies \text{X} : f(g) \qquad (<)$$

Right application (>) applies the lambda expression of a word of type X/Y to that of the word to its right (a Y), while left application (<) , assuming a type of X\Y, applies it to that of the word of type Y to the left. The combination of "the" with "dog" is an example of right application.

The other combinator employed involves composing the lambda expressions.

$$\text{X/Y} : f \quad \text{Y/Z} : g \implies \text{X/Z} : \lambda x.f(g(x)) \qquad (> \text{B})$$

$$\text{Y}\backslash\text{Z} : g \quad \text{X}\backslash\text{Y} : f \implies \text{X}\backslash\text{Z} : f(g(x)) \qquad (< \text{B})$$

Again, there are two directions, left and right composition, which are applicable depending on the syntactic types of the respective words.

Given a lexicon and following these rules, the model can produce parses for whole sentences such as the example in Figure 2.1. Typically, there may be a forest of such trees, consisting of all parses for the sentence that are consistent with the meaning representation. However, to simulate the problem where children do not know the meaning of the sentence, training data consists of sentences paired with a *set* of possible meaning candidates rather than a *single* observed gold meaning. The parse forest for a complete training item, as opposed to a single sentence-meaning pair, is constructed by taking the union of the parses of each pairing of the sentence with its various meaning candidates. Thus, they employ a version of the parsing model described in Kwiatkowski et al. (2010) to parse each sentence-meaning pair individually and combine the results into a single forest.

The probability of a particular parse is computed in much the same way as for a probabilistic context-free grammar (i.e., by weighting productions and computing their product), where a production consists of a local tree (a parent and its immediate children), and its weight defines a conditional probability of the children given their common parent. Defining their model according to this context-free formulation results in a generative joint model of meaning-sentence pairs, and employing a packed forest representation for the union of the parses over all possible sentence-meaning pairs permits the applications of techniques familiar from probabilistic context-free grammar. For instance, the most probable parse can be computed as per CKY, and the model can propose meanings for a sentence by reading off the logical expression associated with this most probable parse. Similarly, the expected counts for each production can be estimated using the inside-outside algorithm, which they incorporate into an online parameter estimation algorithm that makes a single pass over the training data and updates the production weights after each sentence.

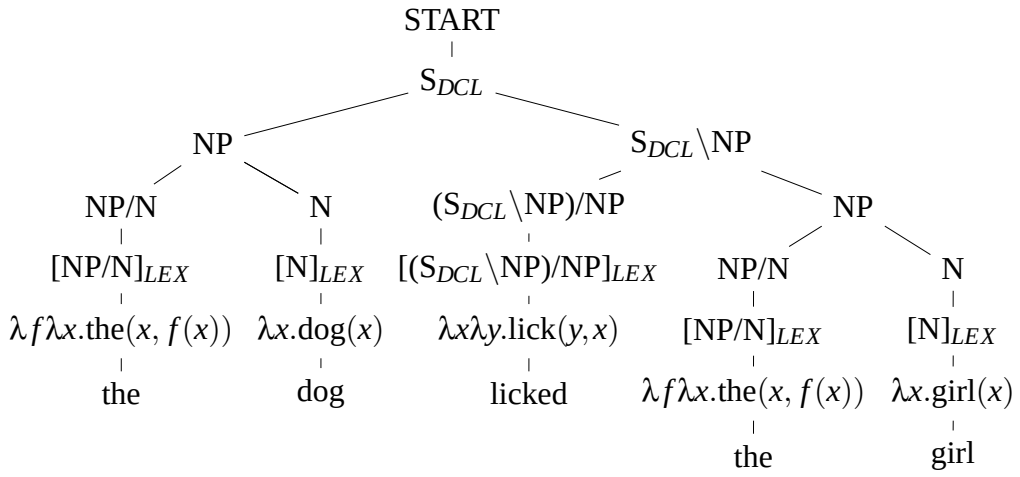

Figure 2.1: A CCG-based parse tree for the sentence "the dog licked the girl" with meaning representation $\text{lick}(\text{the}(x,\text{dog}(x)),\text{the}(y,\text{girl}(y)))$.

The lexicon itself is populated according to a procedure that enumerates all the possible decompositions according to the combinators and maps the individual sub-expressions of the decompositions to words of the accompanying sentence. At the heart of this procedure is a *split* function which takes a CCG category X paired with a fragment h of the meaning representation and enumerates all possible ways h can be split into two sub-expression f and g with syntactic categories C_{LEFT} and C_{RIGHT} which recombine according to application or composition to recover the original X : h pair. In practice, f and g must be constrained since there are technically an infinite number of ways to decompose h. In particular, there are three restrictions.

- **No Vacuous Variables**: All variables that appear as arguments must also appear in the body of the lambda expression. That is, if f is of the form $\lambda x.e$, where e is a logical expression, e must contain variable x.

- **Limited Application**: f cannot contain meta-variables (i.e., variables that stand in for functions) that are applied to non-variable sub-expressions of h. This restriction forbids decompositions such as:

$$h = \lambda x.\text{lick}(x,\text{Sam})$$
$$f = \lambda q.q(\text{Sam})$$
$$g = \lambda y\lambda x.\text{lick}(x,y).$$

Algorithm 1 The parsing algorithm of Kwiatkowski et al. (2012) with a single observed meaning, takes a sentence paired with a meaning and visits each span of the sentence in a fashion similar to a top down variant of CKY, assigning a syntactic category and meaning fragment to each.

function PARSE(sentence $w_1, ...w_n$, meaning representation m)
  Ch = the parse chart as an $n \times n$ matrix of category-meaning pair sets
  Ch[1][n-1] $\leftarrow$ C : m_t ▷ where C is the top-level category of the parse
  for $i = n, ..., 2, j = 1...(n-i)+1$ **do**
    for X : $h \in$ Ch[j][i] **do**
      for $(\mathrm{C}_L : m_L, \mathrm{C}_R : m_R) \in split(\mathrm{X} : h)$ **do**
        for $k = 1, ...i-1$ **do**
          Ch[j][k] $\leftarrow \mathrm{C}_L : m_L$
          Ch[$j+k$][$i-k$] $\leftarrow \mathrm{C}_R : m_R$
  return Ch

- **Limited Coordination Extraction**: g cannot contain more than some number N of the conjuncts in h. For instance, if h is $\text{lick}(x,y) \wedge \text{the}(x, \text{dog}(x)) \wedge \text{the}(y, \text{girl}(y))$, a set of three conjuncts, and $N = 1$, g could only contain one of $\text{lick}(x,y)$, $\text{the}(x, \text{dog}(x))$, or $\text{the}(y, \text{girl}(y))$.

The first two restrictions guarantee at most an exponential number of decompositions, while the third further restricts this to a polynomial of degree N. That is, if the meaning m contains $|m|$ conjuncts, the number of possible splits at each node in the parse tree is at most $O(|m|^N)$.

The parsing procedure is outlined in Algorithm 1. Assuming a single meaning m, the procedure starts by applying *split* to $h = m$, and then populates the parse chart by repeatedly applying *split* to the successively smaller sub-expression while simultaneously matching them up with spans of the sentence. The algorithm resembles CKY but where the grammar rules are generated on the fly from the meaning representation. If there are $|w|$ words in sentence w, parsing a single sentence-meaning pair $\langle w, m \rangle$ has an upper bound of $O(|w|^3 \cdot |m|^N)$, where N is an arbitrary parameter specifying the maximum number of conjuncts of the meaning that can be matched with a single rule and is chosen manually to balance the tradeoff between complexity and accuracy. If there are M possible meaning candidates in the scene, this becomes $O(M \cdot |w|^3 \cdot |m|^N)$. In particular, Kwiatkowski et al. (2010) cites an N of 4 and Kwiatkowski et al. (2012)

an M of 7 in their experiments. If one assumes that the number of conjuncts within the meaning representation is roughly proportional to the number of words in the sentence, this essentially means that parsing requires running an $O(|w|^7)$ algorithm for each meaning candidate in the scene for each sentence.

As a syntax-aware word learner, the model incorporates elements of both syntactic bootstrapping and semantic bootstrapping, i.e., the leveraging of syntactic informtion for acquiring word meanings and the leveraging of semantic information in the acquisition of syntax. For instance, by learning lexical entries like the one for "licked" in our example which expects a NP to the right, the model can theoretically narrow down the set of words that are likely to map to the object of the verb. In fact, they showed that the model preferred an SVO order for transitive constructions when trained on a an English corpus of child-directed speech.

This acquisition of syntactic categories, in turn, is jump-started by the deterministic assignment of basic atomic syntactic categories such as N, NP, and PP based on a small set of templates. For example, if a lexical entries meaning representation component consists of a single unary predicate such as $\lambda x.\text{dog}(x)$, the model deterministically assigns a category of N.

Being based on a grammatical formalism that tightly couples semantics and syntax, the model elegantly incorporates syntactic and semantic learning in a single joint model. Indeed they showed that the model learned to assign high probability to the gold meaning of words, and argued that syntactic bootstrapping helped the most in the case of infrequent words, possibly simulating the role of syntax in fast mapping.

The model is similar in several respects to the novel model we present in Chapter 8, but there are limitations that make it less suitable for our purposes. The most significant issue is the constraints on referential ambiguity in their experiments, i.e., a maximum of seven meaning candidates per scene, since this makes it impossible to judge the learning performance of the model, and, thus, the effect of syntax, at higher, and possibly more realistic, levels of ambiguity. These constraints on ambiguity are due in large part to the sheer computational complexity of parsing, which is roughly $O(|w|^{N+3})$ for a single meaning candidate and increasing by an additional $O(|w|^{N+3})$ for each additional candidate meaning, since the system must repeatedly parse the same sentence multiple times, once per candidate. Thus, a significant amount of the limiting complexity arises from the linear growth in complexity per meaning candidate, but also from the flexibility allowed to the system to explore alternative decompositions of the meaning representation. We will address both of these issues in our choice of

model and parsing algorithms laid out primarily in Chapters 4 and 8. At the same time, we intend to address a few less significant points. For one, it is difficult to assess the exact contribution of syntax since they do not present a comparison with other syntax-unaware word learners. Secondly, the meaning representations of their corpus is automatically constructed from syntactic dependency analyses, possibly resulting in an artificially tight syntax-semantics match, which would boost performance for their syntactic model, something that will be addressed by the corpus introduced in Chapter 7.

Chapter 3

Grammar Background

This chapter covers the background in formal language theory necessary for understanding the semantic parsing and word learning models presented in Chapters 6 and 8. We begin with a brief overview of context-free grammar for string languages, where we establish some notation and otherwise set the stage for the later sections that may be less familiar on graph and synchronous graph-string grammars. The chapter is primarily review, extracted from existing literature, but there are novel elements introduced in Section 3.3.1 on a minor extension to synchronous grammar and Section 3.4 on a generalization of probabilistic grammar where rules are not only associated with a single weight, but also a factorization of that weight.

3.1 Context-free grammar

A context-free grammar (CFG) can be formally defined by a tuple $\langle \Sigma, \mathcal{N}, S, \mathcal{R} \rangle$.

- Σ is a an alphabet of terminal symbols.
- $\mathcal{N} \subset \Sigma$ is an alphabet of nonterminal symbols.
- $S \in \mathcal{N}$ is a nonterminal specially designated as the start symbol of the grammar from which all derivations begin.
- $\mathcal{R}$ is a set of productions of the form $A \to x$ where A is a nonterminal and $x \in (\Sigma \cup \mathcal{N})^*$ is an arbitrary string of terminal and nonterminal symbols.

Figure 3.1 illustrates a simple CFG which generates sentences such as

(3.1) The dog licked the girl as she sternly scolded him.

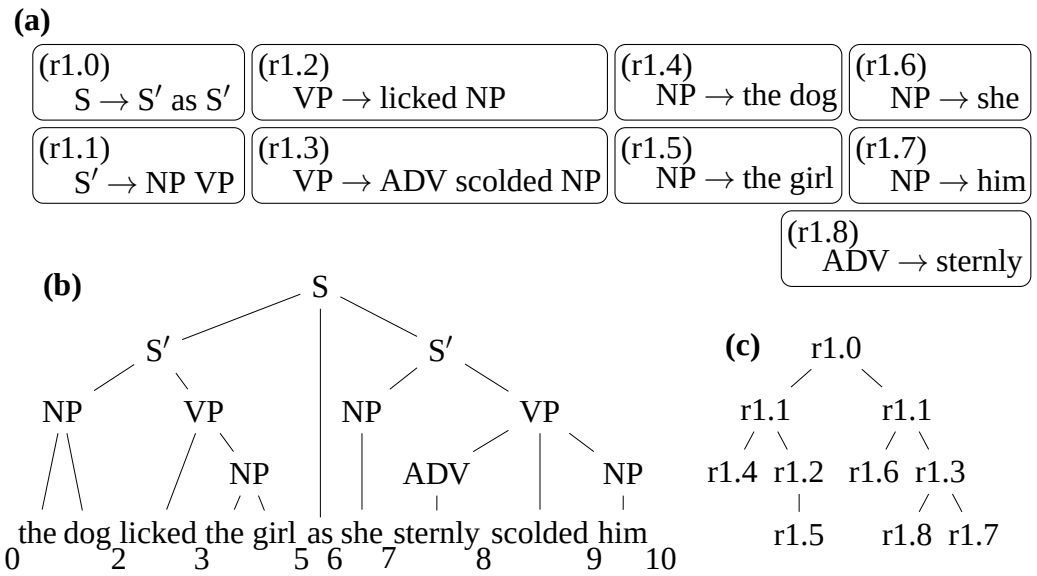

Figure 3.1: (a) A CFG with start symbol S, (b) a parse tree, and (c) the equivalent derivation tree representing all possible derivations of an example sentence. Words are numbered for ease of reference in parsing discussion in Section 3.3.2.

Context-free grammar is a type of string rewriting formalism where strings are gradually built up by replacing nonterminal symbols by their expansions according to the right hand sides of rules. A full derivation consists of a sequence of such rule applications, repeated until we arrive at a string consisting entirely of terminal symbols. As an example, one possible derivation for sentence 3.1 under the grammar of Figure 3.1 might begin with the following two steps

$$\mathrm{S} \overset{r1.0}{\Rightarrow} \mathrm{S}' \text{ as } \mathrm{S}' \overset{r1.1}{\Rightarrow} \mathrm{NP\ VP \text{ as } S}'$$

where $x \overset{r}{\Rightarrow} y$ denotes that string y can be arrived at by applying rule r to rewrite some occurrence of the symbol on the left hand side of rule r in string x. The process is nondeterministic since, for example, there is nothing specifying that a derivation must first expand the left-most occurrence of S′ above before the right occurrence. This nondeterminism can be explicitly represented in a parse tree such as shown in Figure 3.1(b) or an equivalent derivation tree as in Figure 3.1(c). Derivation trees are essentially the same as parse trees where the relationship to rule applications is made more explicit.

3.2 Hyperedge replacement grammar

Hyperedge Replacement Grammar (HRG) is a generalization of CFG to graph languages (see Drewes et al. (1997) for an overview). Where a CFG builds up *strings* by replacing *symbols* with new *substrings*, an HRG builds *graphs* by replacing *edges* with *subgraphs*. Although there has been a recent surge in interest in graph grammar, driven by the availability of corpora for representing linguistic meaning with graphs, HRG is a relatively new introduction and current empirical work is still at a fairly preliminary stage (Jones et al., 2012a; Braune et al., 2014). However, HRG has been formally studied by theoretical computer scientists for much longer (Feder, 1971; Pavlidis, 1972), increasing its appeal since the properties of HRG are already well explored and there are many existing algorithms for working with them. Furthermore, the similarity to CFG makes HRG particularly interesting for natural language processing, because it is easier to adapt and integrate with the prolific literature on syntactic parsing with CFG than many more distantly related formalisms might be.

That said, parsing with HRG is expensive. In fact, in the most general form, parsing is NP complete (Drewes et al., 1997). Of course, there are known polynomial time algorithms for parsing relatively general classes of HRG, but the computational expense can still be prohibitive, depending on the application. The parsing algorithm of Chiang et al. (2013), for instance, takes time exponential in the degree and the tree width of the graph, both of which closely relate to the density of the graph. However, Chapter 4 describes a novel algorithm that operates specifically on tree-shaped graphs which takes time linear in the size of the tree.

The un-forestized scene graphs of the Frog Stories described in Chapter 7 which we use for our word learning experiments are fairly dense, making them expensive to handle. Instead, we work with a special case of HRG for generating unordered trees to work with the forestized scenes described in Section 7.4. However, there are few context-free grammar-based formalisms for describing unordered trees that are as well studied as HRG. Furthermore, much of the work we describe such as the optimizations in the parsing algorithm in 4 generalize to more expressive graphs. Additionally, there is growing interest in applying graph formalisms to problems in natural language processing, and it is our hope that adopting the same standard terminology used in both these forays into applied HRG as well as that of the theoretical literature will make it easier to draw connections. Thus, although we work primarily with a restriction, we define HRG in full.

We start by defining a hypergraph, a generalization of a graph where edges may link any finite number of vertices. Typically, such edges are called *hyperedges*, but we will use the terms edge and hyperedge interchangeably. Formally, a hypergraph is a tuple $\langle \mathcal{V}, \mathcal{E}, a, \ell, x \rangle$.

- $\mathcal{V}$ and $\mathcal{E}$ are finite sets of vertices and hyperedges, respectively.
- The *attachment function*, $a : \mathcal{E} \to \mathcal{V}^*$, maps each hyperedge $e \in \mathcal{E}$ to a sequence of pairwise distinct vertices from $\mathcal{V}$, where we call the length of the vertex sequence $a(e)$ the *arity* of edge e.
- The *labeling function*, $\ell : \mathcal{E} \to \Sigma$, maps each hyperedge to a symbol in some ranked alphabet Σ, where the rank of the label symbol $\ell(e)$ is the same as the arity of edge e.
- Finally, each graph has a set of zero or more *external vertices*, arranged in a sequence $x \in \mathcal{V}^*$, where the vertices are pairwise distinct just as those of a hyperedge. As another point of similarity with hyperedges, hypergraphs also have an arity, defined as the number of its external vertices (i.e., the length of x). There is a strong parallel between the external vertices of hypergraphs and the vertices of hyperedges, a fact that, as we shall see, plays a crucial role in the edge rewriting mechanism of HRG.

Observant readers may notice that while function ℓ labels edges, there is no such function for labeling vertices. In fact, vertices are unlabeled, but labels can be simulated by treating unary hyperedges (i.e., hyperedges with a single vertex) as vertex labels.

While edges can link an arbitrary number of vertices, we are primarily interested in languages of simple directed graphs, hypergraphs where each edge is either binary or, for vertex labels, unary. In this case, we can indicate visually the ordering on a binary edge with vertex sequence $v_0 v_1$ by an arrow pointing from vertex v_0 to v_1. The graph at the bottom right of Figure 3.2(b), is an example of a graph, similar to the scene graphs described in Chapter 7. In fact, we can apply the same conventions introduced in Section 7.3 for translating between the language of predicate calculus expressions and graphs to arrive at the following:

(3.2) $\text{lick}(e_1) \wedge \text{agent}(e_1, x_1) \wedge \text{dog}(x_1) \wedge \text{theme}(e_1, x_2) \wedge \text{girl}(x_2) \wedge$
$\text{scold}(e_2) \wedge \text{agent}(e_2, x_2) \wedge \text{theme}(e_2, x_1) \wedge \text{stern}(e_2)$

The graph representation has unary edges for expressing entities such as *girl* and *dog* and predicates like *lick* and *scold* and binary edges with labels like *agent* for specifying the thematic relations in events. Additionally, while vertices are unlabeled in the formal definition of hypergraphs, we have also included labels to uniquely identify vertices in the figure, simply to facilitate discussion and to make it clearer how the graph relates to the logical expression in example 3.2 above.

We now describe hyperedge replacement grammar, an edge rewriting system for generating hypergraphs, formally defined in a manner similar to CFG as a tuple $\langle \Sigma, \mathcal{N}, S, \mathcal{R} \rangle$.

- Σ is a ranked alphabet of terminal symbols (i.e, a set of symbols where each symbol is associated with some integer greater than or equal to zero identifying its *rank*).
- $\mathcal{N}$ is a ranked alphabet of nonterminal symbols.
- $S \in \mathcal{N}$ is the start symbol.
- $\mathcal{R}$ is a finite set of rules of the form $A \to h$, where h is a hypergraph with edge labels from $\Sigma \cup \mathcal{N}$ and $A \in \mathcal{N}$ has rank equal to the arity of h.

Figure 3.2 shows an example of an HRG and a sample derivation. The external vertices of the right-hand side graphs have been shaded and numbered according to their order, while other vertices (the internal vertices) such as in rule $r2.1$ have been left unlabeled. Edges named with nonterminal symbols are dashed to make them easier to identify. Sometimes we will refer to such edges that are labeled with nonterminal symbols as *nonterminal edges*.

Hyperedge replacement, the basic rewriting mechanism of HRG, is an operation which substitutes an entire hypergraph for a single edge. If g is a hypergraph containing edge e, and h is another hypergraph with the same arity as e (i.e., with the same number of external vertices as e has vertices), edge e can be replaced with h by first removing e from g and then "fusing" h and g together at the external vertices of h and the vertices of the sequence $a(e)$. So, if $a(e) = v_0 v_1 ... v_k$ and h has external vertices $u_0 u_1 ... u_k$, we would fuse each u_i to the corresponding v_i.

Much like with CFG where a derivation begins with a string consisting of a single instance of the start symbol and proceeds by successively replacing nonterminals with substrings, derivations under HRG begin with a single edge (with arity equal to the rank of the start symbol) and each subsequent step replaces a nonterminal edge with the right-hand side graph of some rule with a matching left-hand side. For example,

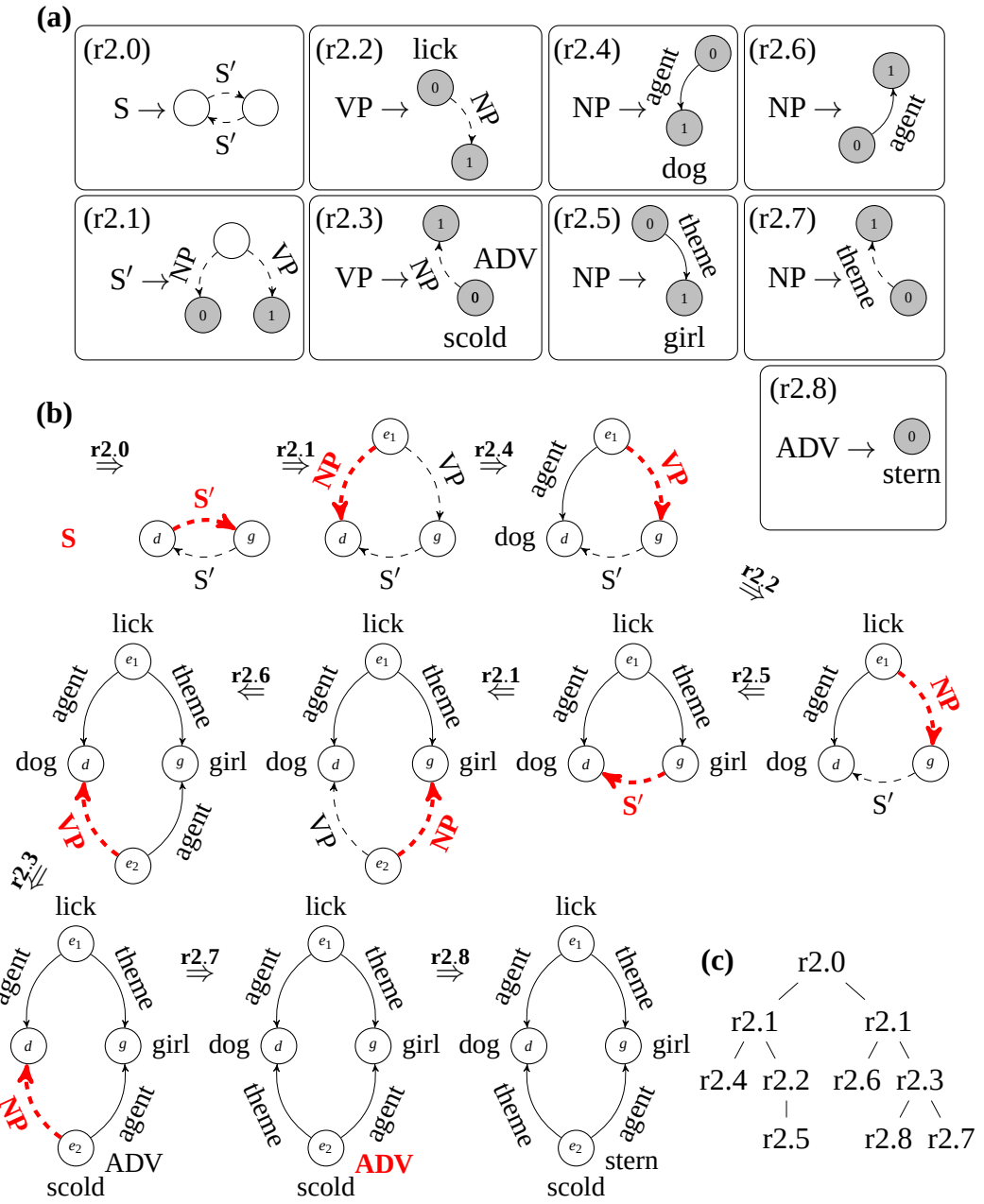

Figure 3.2: (a) An HRG with start symbol S, (b) a particular derivation under the grammar of a graphical representation of a sample meaning representation, and (c) a derivation tree representing all possible derivations. External vertices are shaded and numbered according to their order and nonterminal edges are dashed, while in the derivation the nonterminal edge being replaced is highlighted in red.

in the application of rule $r2.1$ in the second step of the derivation in Figure 3.2(b), the edge $d \xrightarrow{S'} g$ is replaced by the graph $0 \xleftarrow{NP} \circ \xrightarrow{VP} 1$ by removing the S′ edge and then attaching the new subgraph by identifying vertices d and g with external vertices 0 and

1, respectively. Note that the ordering of d and g in edge $d \xrightarrow{\mathrm{S}'} g$ and the ordering of the external vertices in $r2.1$ fully specifies exactly which vertices are "fused" during the replacement operation.

An HRG is context-free in the sense that whether a given rule can be applied at any given stage of a derivation only depends on whether the nonterminal on the left-hand side (and its rank) matches some isolated nonterminal of the intermediate hypergraph derivation, exactly as for string derivations under CFG. Consequently, the order in which rules are applied during a derivation does not impact the set of possible expansions of any of the remaining nonterminals. Again, just as with CFG, this nondeterminism can be represented explicitly by a derivation tree, as illustrated in Figure 3.2(c).

As a notational convenience, we can represent graphs using the language of predicate logic, where vertices are identified with variables. As a consequence, edges, which are labeled tuples of vertices, can be treated as relations, themselves merely tuples of variables. A hypergraph can then be considered to be a logical conjunction of relations. Following this convention, we can translate the final graph in the HRG derivation in Figure 3.2(b) into the logical expression:

$$\begin{aligned}&\mathrm{like}(e_1) \wedge \mathrm{theme}(e_1, g) \wedge \mathrm{girl}(g) \wedge \mathrm{agent}(e_1, d) \wedge \mathrm{dog}(d) \\ &\wedge \mathrm{scold}(e_2) \wedge \mathrm{agent}(e_2, g) \wedge \mathrm{theme}(e_2, d) \wedge \mathrm{stern}(e_2).\end{aligned}$$

For completeness, one can assume that all variables are existentially quantified, though we will generally omit these existential quantifiers for the sake of brevity.

Thus, one can reinterpret the grammar in Figure 3.2(a) as operating directly on logical expressions.

$$\mathrm{S} \to \mathrm{S}'(d, g) \wedge \mathrm{S}'(g, d) \tag{r2.0}$$

$$\mathrm{S}' \to \lambda d, g.\mathrm{NP}(e, d) \wedge \mathrm{VP}(e, g) \tag{r2.1}$$

$$\mathrm{VP} \to \lambda e, g.\mathrm{lick}(e) \wedge \mathrm{NP}(e, g) \tag{r2.2}$$

$$\mathrm{VP} \to \lambda e, d.\mathrm{scold}(e) \wedge \mathrm{ADV}(e) \wedge \mathrm{NP}(e, d) \tag{r2.3}$$

$$\mathrm{NP} \to \lambda e, d.\mathrm{agent}(e, d) \wedge \mathrm{dog}(d) \tag{r2.4}$$

$$\mathrm{NP} \to \lambda e, g.\mathrm{theme}(e, g) \wedge \mathrm{girl}(g) \tag{r2.5}$$

$$\mathrm{NP} \to \lambda e, g.\mathrm{agent}(e, g) \tag{r2.6}$$

$$\mathrm{NP} \to \lambda e, d.\mathrm{theme}(e, d) \tag{r2.7}$$

$$\mathrm{ADV} \to \lambda e.\mathrm{stern}(e) \tag{r2.8}$$

Here, we have extended the logical language with lambda expressions in order to specify the external vertices of the right-hand side graphs. In fact, drawing on this analogy between external vertices and the lambda arguments presents another way of visualizing the edge replacement operation, as a substitution operation where nonterminal edges are treated as metavariables. For example, the first application of $r2.1$ in the example derivation involves replacing the first instance of the S' nonterminal in the intermediate graph with the right hand side of the rule.

$$\begin{aligned}(\mathrm{S}'_0(d,g) \wedge \mathrm{S}'_1(g,d))\left[\mathrm{S}'_0 := \lambda d_0, g_0.\mathrm{NP}(e,d_0) \wedge \mathrm{VP}(e,g_0)\right] \\ = (\lambda d_0, g_0.\mathrm{NP}(e,d_0) \wedge \mathrm{VP}(e,g_0))(d,g) \wedge \mathrm{S}'(g,d) \\ = \mathrm{NP}(e,d) \wedge \mathrm{VP}(e,g) \wedge \mathrm{S}'(g,d).\end{aligned}$$

After the substitution, the function is applied to arguments d and g, resulting in a beta reduction and a second substitution where $d_0 := d$ and $g_0 := g$. This parallel between edge replacement and substitution in lambda calculus draws a closer tie with linguistic theories of Montague semantics and the syntax-semantics interface, something that should become clearer in Section 3.3 when we discuss synchronous grammar. In fact, because the logical notation is relatively concise and we expect the audience may be somewhat more familiar with lambda calculus than with edge replacement, we will usually rely on this notation and only return to explicit graphical representations when it is useful for discussing properties that are more easily described in terms of graph theory.

As mentioned in the introductory paragraphs of this section, although HRG is designed to describe languages of arbitrary hypergraphs, we focus on HRG that generate trees. This tree assumption is a common theme in previous parsing work. For instance, work involving semantic dependency graphs often rely on minimum spanning tree algorithms (McDonald et al., 2005), an approach which has been adapted to more general semantic graph structures (Flanigan et al., 2014). In fact, since the parses of HRG are themselves trees, it is possible to identify every HRG parse with a particular tree decomposition of a graph (Lautemann, 1988). Thus, tree shaped representations of more general graphs seem to be a fundamental feature of grammars with context-free derivations. The difference in our work here is that while in the general case there are many such possible tree decompositions and a parser must somehow search over this set to identify a particular one, we assume there is a single such decomposition and enforce the assumption in a preprocessing step applied to the corpus as described in Section 7.4. This decision to enforce a single tree is mainly driven by parsing efficiency

concerns, since parsing with arbitrary HRGs can be highly expensive.

Unordered trees differ in one important respect from the ordered trees that most readers will be familiar with from syntactic parsing. The order of, for instance, the *agent* and *patient* under a *disturb* event is indeterminate. The ordering is a matter of how the particular language realizes such semantic elements in the syntax, but not something to encode in the meaning representation itself. This lack of ordering corresponds to the commutative property of predicate calculus, since

$$\text{disturb}(e) \wedge \text{agent}(e,x_1) \wedge \text{boy}(x_1) \wedge \text{patient}(e,x_2) \wedge \text{owl}(x_2)$$

is true in exactly the same circumstances that a slightly reordered expression is true:

$$\text{disturb}(e) \wedge \text{patient}(e,x_2) \wedge \text{owl}(x_2) \wedge \text{agent}(e,x_1) \wedge \text{boy}(x_1).$$

This lack of ordering is something that is impossible to capture using formalisms based on ordinary CFGs or regular tree grammars, but straightforward using an HRG. Lack of ordering plays a key role in parsing, an issue we will return to in Chapter 4.

Even if we do not make full use of the expressive power of HRG and restrict consideration to tree-shaped graphs, there are still key advantages over some other tree grammars such as regular tree grammar (RTG). For one thing, because there is already a more general formalism, it may be easier to generalize from the tree case to other varieties of graph in future work. Furthermore, unlike strings, directed graphs only define a partial ordering on their constituent vertices, so the trees our grammars generate will also be only partially ordered. This is in contrast to the trees familiar from syntactic parsing, where sibling nodes cannot be reordered without changing the sentence (and, consequently, the syntax tree). However, the tree-shaped meaning representations we will deal with are intended to represent logical structures where order does not matter. For an example, consider the following logical expression, which corresponds to a tree-shaped graph:

$$\text{lick}(e) \wedge \text{agent}(e,x_1) \wedge \text{dog}(x_1) \wedge \text{theme}(e,x_2) \wedge \text{girl}(x_2).$$

It would be undesirable if our grammar treated the following as an entirely different expression since it is logically equivalent and evaluates to the same truth values under the same conditions:

$$\text{lick}(e) \wedge \text{theme}(e,x_2) \wedge \text{girl}(x_2) \wedge \text{agent}(e,x_1) \wedge \text{dog}(x_1).$$

This is an advantage of using a formalism such as graph grammars which is capable of generating and parsing unordered structures. An RTG, for instance, suffers from the

same ordering constraint as a CFG, where different orderings are treated as completely different trees, resulting in different parses. Unlike RTG or other formalisms that only work with ordered trees, however, any HRG would assign the exact same set of parses to both expressions, drawing no distinction.

3.3 Synchronous grammar

A synchronous grammar is a formalism that simultaneously generates items from two or more languages, implicitly defining a relation between the languages. In principle, the languages could be of strings, trees, or arbitrary graphs. Synchronous grammars are closely related to tree transducers, which define relations between tree languages (Shieber, 2004; 2014). Synchronous grammars and tree transducers have both figured prominently in syntax-based machine translation, where they are used to map between the syntax of a source language to that of some target language (Nesson et al., 2006; DeNeefe and Knight, 2009; Yamada and Knight, 2001). We apply synchronous grammar to semantic parsing, which can be seen as a kind of translation problem between natural language and meaning representation, and to modeling the word learning problem. Synchronous grammars have already been applied to the problem of modeling the syntax-semantics interface (Nesson and Shieber, 2006; Han and Hedberg, 2008) and semantic parsing (Wong and Mooney, 2007), and other semantic parsing models such as Lu et al. (2008) or Kate and Mooney (2006), though not explicitly expressed in terms of synchronous grammar, can also be re-interpreted as such. In fact, chapter 6 makes this last assertion clearer by explicitly re-implementing the hybrid tree model of Lu et al. (2008) as a synchronous grammar. Our greatest departure from previous work is to integrate HRG into the synchronous grammar framework to better model commutative meaning languages (i.e., unordered trees), where previous synchronous grammar-based models assume an ordered meaning representation language.

Synchronous grammars can be thought of as a kind of amalgam of two or more different monolingual grammars, produced by binding together rules from these sub-grammars. During derivations, rules are applied in lock-step fashion to produce parallel derivations and yielding tuples of items from the sub-grammars' respective languages. These tuples thereby define a relation. For a more formal description, we simplify by defining synchronous grammars that are restricted to binary relations (i.e., that map between two languages), but it is straightforward to generalize the definition to include grammars for trinary or higher order relations. Assume there are already

grammars with rule sets $\mathcal{R}_0$ and $\mathcal{R}_1$. Then the rules of the synchronous grammar are tuples of the form $\langle r_0, r_1, \frown \rangle$ where $r_0 \in \mathcal{R}_0$ and $r_1 \in \mathcal{R}_1$ and $\frown$ is a bijection defining a one-to-one correspondence between the nonterminals of the right-hand sides of the monolingual rules r_0 and r_1. During derivations, rules expand pairs of nonterminals, where the legal nonterminal pairs are dictated by the bijection $\frown$.

For example, we can bind together the sample HRG and CFG in Figures 3.2 and 3.1 into a single synchronous graph-string grammar by defining rules such as

$$\langle\ \mathrm{S}' \rightarrow \lambda d, g.\mathrm{NP}_{\boxed{0}}(e,d) \wedge \mathrm{VP}_{\boxed{1}}(e,g)\ \big\|\ \mathrm{S}' \rightarrow \mathrm{NP}_{\boxed{0}}\ \mathrm{VP}_{\boxed{1}}\ \rangle$$

where the bijection between nonterminals is indicated by identifying symbols with the same $\boxed{i}$ indices. During a derivation these monolingual rule pairs are applied simultaneously to expand corresponding nonterminals in parallel. Consider how this synchronous rule might be applied to the following intermediate stage of a synchronous derivation:

$$\langle\ \mathrm{S}'_{\boxed{0}}(d,g) \wedge \mathrm{S}'_{\boxed{1}}(g,d)\ \big\|\ \mathrm{S}'_{\boxed{0}}\ \text{as}\ \mathrm{S}'_{\boxed{1}}\ \rangle.$$

There are two choices for nonterminals that the rule can expand: either the pair $\langle \mathrm{S}'_{\boxed{0}}, \mathrm{S}'_{\boxed{0}} \rangle$ or the pair $\langle \mathrm{S}'_{\boxed{1}}, \mathrm{S}'_{\boxed{1}} \rangle$. However, the bijection does not permit the expansion of pairs $\langle \mathrm{S}'_{\boxed{0}}, \mathrm{S}'_{\boxed{1}} \rangle$ or $\langle \mathrm{S}'_{\boxed{1}}, \mathrm{S}'_{\boxed{0}} \rangle$, because there is no map between those nonterminal pairs (as indicated by the mismatching indices). Thus, either of the following are legal next steps in the derivation:

$$\langle\ \mathrm{NP}_{\boxed{0}}(e,d) \wedge \mathrm{VP}_{\boxed{1}}(e,g) \wedge \mathrm{S}'_{\boxed{2}}(g,d)\ \big\|\ \mathrm{NP}_{\boxed{0}}\ \mathrm{VP}_{\boxed{1}}\ \text{as}\ \mathrm{S}'_{\boxed{2}}\ \rangle$$

or

$$\langle\ \mathrm{S}'_{\boxed{0}}(g,d) \wedge \mathrm{NP}_{\boxed{1}}(e,d) \wedge \mathrm{VP}_{\boxed{2}}(e,g)\ \big\|\ \mathrm{S}'_{\boxed{0}}\ \text{as}\ \mathrm{NP}_{\boxed{1}}\ \mathrm{VP}_{\boxed{2}}\ \rangle,$$

but not

$$\langle\ \mathrm{NP}_{\boxed{0}}(e,d) \wedge \mathrm{VP}_{\boxed{1}}(e,g) \wedge \mathrm{S}'_{\boxed{2}}(g,d)\ \big\|\ \mathrm{S}'_{\boxed{0}}\ \text{as}\ \mathrm{NP}_{\boxed{1}}\ \mathrm{VP}_{\boxed{2}}\ \rangle.$$

In practice, we will assume that only nonterminals of the same symbol map to one another, permitting a more compact representation for synchronous rules:

$$\mathrm{S}' \rightarrow \langle\ \lambda d, g.\mathrm{NP}_{\boxed{0}}(e,d) \wedge \mathrm{VP}_{\boxed{1}}(e,g)\ \big\|\ \mathrm{NP}_{\boxed{0}}\ \mathrm{VP}_{\boxed{1}}\ \rangle \qquad \text{(r3.1)}$$

where we can use a single left-hand side symbol since both of the sub-rules are guaranteed to have the same left-hand side (modulo symbol rank, which can be inferred from

$$\mathrm{S} \rightarrow \langle\ \mathrm{S}'_{\boxed{0}}(d,g) \wedge \mathrm{S}'_{\boxed{1}}(g,d)\ \|\ \mathrm{S}'_{\boxed{0}}\ \text{as}\ \mathrm{S}'_{\boxed{1}}\ \rangle \quad \text{(r3.0)}$$

$$\mathrm{S}' \rightarrow \langle\ \lambda d,g.\mathrm{NP}_{\boxed{0}}(e,d) \wedge \mathrm{VP}_{\boxed{1}}(e,g)\ \|\ \mathrm{NP}_{\boxed{0}}\ \mathrm{VP}_{\boxed{1}}\ \rangle \quad \text{(r3.1)}$$

$$\mathrm{VP} \rightarrow \langle\ \lambda e,g.\mathrm{lick}(e) \wedge \mathrm{NP}_{\boxed{0}}(e,g)\ \|\ \text{licked}\ \mathrm{NP}_{\boxed{0}}\ \rangle \quad \text{(r3.2)}$$

$$\mathrm{VP} \rightarrow \langle\ \lambda e,d.\mathrm{scold}(e) \wedge \mathrm{ADV}_{\boxed{0}}(e) \wedge \mathrm{NP}_{\boxed{1}}(e,d)\ \|\ \mathrm{ADV}_{\boxed{0}}\ \text{scolded}\ \mathrm{NP}_{\boxed{1}}\ \rangle \quad \text{(r3.3)}$$

$$\mathrm{NP} \rightarrow \langle\ \lambda e,d.\mathrm{agent}(e,d) \wedge \mathrm{dog}(d)\ \|\ \text{the dog}\ \rangle \quad \text{(r3.4)}$$

$$\mathrm{NP} \rightarrow \langle\ \lambda e,g.\mathrm{theme}(e,g) \wedge \mathrm{girl}(g)\ \|\ \text{the girl}\ \rangle \quad \text{(r3.5)}$$

$$\mathrm{NP} \rightarrow \langle\ \lambda e,g.\mathrm{agent}(e,g)\ \|\ \text{she}\ \rangle \quad \text{(r3.6)}$$

$$\mathrm{NP} \rightarrow \langle\ \lambda e,d.\mathrm{theme}(e,d)\ \|\ \text{him}\ \rangle \quad \text{(r3.7)}$$

$$\mathrm{ADV} \rightarrow \langle\ \lambda e.\mathrm{stern}(e)\ \|\ \text{sternly}\ \rangle \quad \text{(r3.8)}$$

$$\mathrm{VP} \rightarrow \langle\ \lambda e,d.\mathrm{scold}(e) \wedge \mathrm{NP}_{\boxed{0}}(e,d)\ \|\ \mathrm{MADV}\ \text{scolded}\ \mathrm{NP}_{\boxed{0}}\ \rangle \quad \text{(r3.9)}$$

$$\mathrm{MADV} \rightarrow \langle\ -\ \|\ \text{sternly}\ \rangle \quad \text{(r3.10)}$$

Table 3.1: A synchronous grammar that jointly generates ⟨meaning representation, sentence⟩ pairs. Rule $r3.10$ is monolingual, expanding the MADV nonterminal in $r3.9$ without generating anything in the meaning representation, as described in Section 3.3.1.

the right-hand side). Creating synchronous rules from the remaining rules from our example graph and string grammars produces the grammar in Table 3.1. These rules can then be applied according to the derivation tree in Figure 3.2 to simultaneously produce the ⟨meaning representation, sentence⟩ pair:

$$\begin{aligned}\langle\ &\mathrm{lick}(e_1) \wedge \mathrm{agent}(e_1,x_2) \wedge \mathrm{dog}(x_2) \wedge \mathrm{theme}(e_1,x_1) \wedge \mathrm{girl}(x_1) \wedge \\ &\mathrm{scold}(e_2) \wedge \mathrm{agent}(e_2,x_1) \wedge \mathrm{theme}(e_2,x_2) \wedge \mathrm{stern}(e_2)\ \| \\ &\text{the dog licked the girl as she sternly scolded him}\ \rangle.\end{aligned}$$

3.3.1 Adding monolingual rules

With a traditional synchronous grammar the number of nonterminals in each of the linked sub-rules must be exactly equal. Thus, each contribution to one side of the yield during a derivation always accompanies a simultaneous contribution to the others.

What this means in the case of joint semantics-syntax grammars is that, effectively, the grammar would need to decompose the meaning representation and the sentence and pair these up so that every word is assigned meaning and every fragment of meaning is represented in the words.

It may make intuitive sense that all words must be represented in some form in the meaning representation, but only so long as one assumes that the meaning representation fully covers the sentence. The validity of this assumption depends on the corpus and the annotation scheme employed, and it is definitely not the case for most word learning and semantic parsing corpora (and the frog stories corpus is no exception) where there could be any number of words with no obvious contribution to the meaning representation. For instance, sometimes annotations will omit modifiers such as *stern* in our running example.

The conventional lock-step derivation process of a synchronous grammar is not well suited for modeling this sort of situation, but a minor extension could help. In particular, it is possible to relax the bijection between nonterminals, $\frown$, so that instead of requiring it to cover all nonterminals, we can restrict it to some subset, where the remaining nonterminals are monolingual. These monolingual nonterminals must be expanded by special rules that behave like ordinary monolingual grammar rules that only contribute to one side of the yield (words in the sentence but not predicates in the meaning, for example). Thus, in addition to synchronous rules of the form $\langle r_0, r_1, \frown \rangle$, the grammar may also include monolingual rules such as $\langle r_0, -, \emptyset \rangle$ that only expand monolingual nonterminals on the left or $\langle -, r_1, \emptyset \rangle$ which only expand monolingual nonterminals on the right.

The bottom two rules in Table 3.1 illustrate how monolingual rules can be introduced into a synchronous grammar to generate words without counterparts in the meaning representation. Rule $r3.9$ is semi-synchronous, where the MADV nonterminal is monolingual and does not correspond to anything on the meaning representation side. This monolingual nonterminal cannot be expanded by rule $r3.8$ since it expands two nonterminals instead of one. Rather, a derivation must employ the monolingual rule $r3.10$ to expand that portion of the sentence, which has no impact on the meaning representation portion of the yield. By including monolingual rules, the grammar can

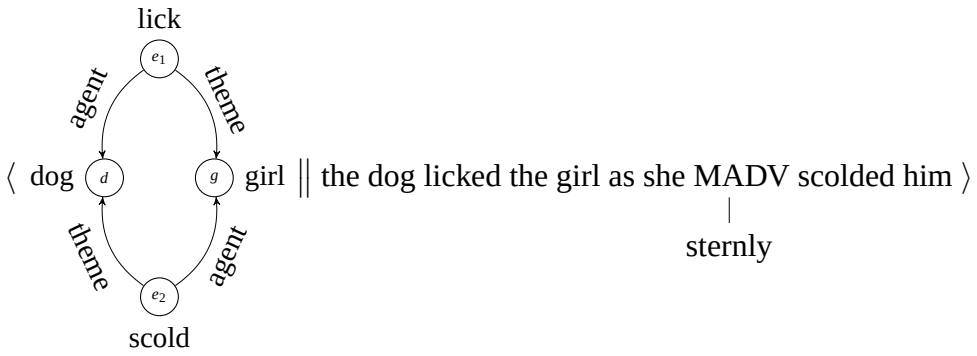

Figure 3.3: A monolingual production in a synchronous derivation. The MADV has no counterpart in the meaning representation but generates the word “sternly” in the sentence.

still generate ⟨meaning representation, sentence⟩ pairs such as

$$\langle\ \text{lick}(e_1) \wedge \text{agent}(e_1, x_2) \wedge \text{dog}(x_2) \wedge \text{theme}(e_1, x_1) \wedge \text{girl}(x_1) \wedge$$
$$\text{scold}(e_2) \wedge \text{agent}(e_2, x_1) \wedge \text{theme}(e_2, x_2)\ \|$$
$$\text{the dog licked the girl as she sternly scolded him}\ \rangle$$

where the meaning representation fails to include the *stern* modifier for the event. However, removing rules $r3.9$ and $r3.10$ would cause a parser to fail.

Note that because synchronous rules can only expand synchronous nonterminals and monolingual rules can only expand monolingual nonterminals, it is impossible once a particular branch of a derivation tree enters a monolingual mode to return to synchronous generation downstream. The entire subtree must be filled out with monolingual rules.

These synchronous grammars with monolingual rules have a close relationship to the top-down tree transducers with regular look-ahead studied by Engelfriet (1977). Transducers with regular look-ahead are able to look indefinitely deep into a left tree to see if it meets some preset condition before deciding whether to apply a rule, where the look-ahead mechanism can be described as an RTG. Essentially, a rule can only be applied at a particular location within the left tree if its subtrees satisfy a membership test in the tree language defined by the look-ahead grammar. In our case, the monolingual rules behave like this look-ahead mechanism where, in parsing, a particular semi-synchronous rule containing monolingual nonterminals can only be included in a valid parse if the monolingual portions of the yield satisfy membership tests defined by

the monolingual portion of the grammar. In terms of the example in Figure 3.3, $r3.9$ only applies if the one word substring "sternly" can be parsed using monolingual rule $r3.10$. One key difference is that while Engelfriet (1977) worked with RTGs, the synchronous grammars with monolingual rules are general to any grammar with context-free derivations. Another important difference is that Engelfriet (1977) only defined look-ahead for one side, but in our case "look-ahead" can be applied on either side or both simultaneously, since rules of either $\langle r, -, \emptyset \rangle$ or $\langle -, r, \emptyset \rangle$ form are permitted, a feature we will find useful when modeling a joint scene-meaning-utterance three-way grammar, where, in addition to modeling the sort of meaning-utterance relationship just discussed, utterances will also tend to only discuss relatively small subsets of the scene as a whole.

3.3.2 Parsing: Training vs. translating

Synchronous grammars can be used in one of two different ways: to test whether a given tuple of items from two or more languages satisfies a particular relation, or to generate translations from one language to another. The first is achieved by parsing tuples synchronously (i.e., memberships checking). The second is achieved by parsing an item from one of the languages using a *projection* of the grammar where only the portion of the rules relating to the item to be translated is used. Either case yields a parse forest which can be interpreted as yet another grammar where rules are based on those from the original grammar that appear in the parses but with nonterminals refined to identify ⟨nonterminal,span⟩ combinations. Table 3.2 presents an example which we will discuss in greater detail momentarily. In the case of synchronous parsing, the grammar has one tuple in its language, while in the second, there is a potentially very large language of tuples where one of the elements of every tuple is constrained to equal the item that was parsed. In the case of translation, taking a projection of the language of tuples can produce a set of possible candidate translations in the target language. Both of these cases will prove useful for our applications. In particular, while training a semantic parser from a corpus of observed meaning representations paired with their corresponding sentences, we will want to use full synchronous parsing to enumerate the ways that the grammar might map meaning representation to sentence. However, at test time semantic parsers are only given the sentence and must translate this into a meaning representation, a case involving projection.

Table 3.2 illustrates an example of a translating grammar, the result of parsing

$$\mathrm{S}^{0}_{10} \rightarrow \langle\ \mathrm{S}'^{0}_{5\boxed{0}}(d,g) \wedge \mathrm{S}'^{6}_{10\boxed{1}}(g,d)\ \|\ \mathrm{S}'^{0}_{5\boxed{0}}\ \text{as}\ \mathrm{S}'^{6}_{10\boxed{1}}\ \rangle \quad \text{(r4.0)}$$

$$\mathrm{S}'^{0}_{5} \rightarrow \langle\ \lambda d,g.\mathrm{NP}^{0}_{2\boxed{0}}(e,d) \wedge \mathrm{VP}^{2}_{5\boxed{1}}(e,g)\ \|\ \mathrm{NP}^{0}_{2\boxed{0}}\ \mathrm{VP}^{2}_{5\boxed{1}}\ \rangle \quad \text{(r4.1a)}$$

$$\mathrm{S}'^{6}_{10} \rightarrow \langle\ \lambda d,g.\mathrm{NP}^{6}_{7\boxed{0}}(e,d) \wedge \mathrm{VP}^{7}_{10\boxed{1}}(e,g)\ \|\ \mathrm{NP}^{6}_{7\boxed{0}}\ \mathrm{VP}^{7}_{10\boxed{1}}\ \rangle \quad \text{(r4.1b)}$$

$$\mathrm{VP}^{2}_{5} \rightarrow \langle\ \lambda e,g.\mathrm{lick}(e) \wedge \mathrm{NP}^{3}_{5\boxed{0}}(e,g)\ \|\ \text{licked}\ \mathrm{NP}^{3}_{5\boxed{0}}\ \rangle \quad \text{(4.r2)}$$

$$\mathrm{VP}^{7}_{10} \rightarrow \langle\ \lambda e,d.\mathrm{scold}(e) \wedge \mathrm{ADV}^{7}_{8\boxed{0}}(e) \wedge \mathrm{NP}^{9}_{10\boxed{1}}(e,d)\ \|\ \mathrm{ADV}^{7}_{8\boxed{0}}\ \text{scolded}\ \mathrm{NP}^{9}_{10\boxed{1}}\ \rangle \quad \text{(r4.3)}$$

$$\mathrm{NP}^{0}_{2} \rightarrow \langle\ \lambda e,d.\mathrm{agent}(e,d) \wedge \mathrm{dog}(d)\ \|\ \text{the dog}\ \rangle \quad \text{(r4.4)}$$

$$\mathrm{NP}^{3}_{5} \rightarrow \langle\ \lambda e,g.\mathrm{theme}(e,g) \wedge \mathrm{girl}(g)\ \|\ \text{the girl}\ \rangle \quad \text{(r4.5)}$$

$$\mathrm{NP}^{6}_{7} \rightarrow \langle\ \lambda e,g.\mathrm{agent}(e,g)\ \|\ \text{she}\ \rangle \quad \text{(r4.6)}$$

$$\mathrm{NP}^{9}_{10} \rightarrow \langle\ \lambda e,d.\mathrm{theme}(e,d)\ \|\ \text{him}\ \rangle \quad \text{(r4.7)}$$

$$\mathrm{ADV}^{7}_{8} \rightarrow \langle\ \lambda e.\mathrm{stern}(e)\ \|\ \text{sternly}\ \rangle \quad \text{(r4.8)}$$

$$\mathrm{VP}^{7}_{10} \rightarrow \langle\ \lambda e,d.\mathrm{scold}(e) \wedge \mathrm{NP}^{9}_{10\boxed{0}}(e,d)\ \|\ \mathrm{MADV}^{7}_{8}\ \text{scolded}\ \mathrm{NP}^{9}_{10\boxed{0}}\ \rangle \quad \text{(r4.9)}$$

$$\mathrm{MADV}^{7}_{8} \rightarrow \langle\ -\ \|\ \text{sternly}\ \rangle \quad \text{(r4.10)}$$

Table 3.2: The synchronous grammar that encodes the set of possible translations of the string in Figure 3.1 using the grammar of Table 3.1. There are two possible meaning candidates, one that includes the *stern* modifier, the other that omits it, depending on whether rule $r4.3$ or $r4.9$ is used for the second VP.

the sentence in Figure 3.1 with the synchronous grammar in Table 3.1. By marking nonterminals with the span they dominate in the string, the grammar enforces that the string portion of the pairs it generates must be the same as that of Figure 3.1. However, the meaning representation is less constrained, since there are the monolingual rules that permit the optional omission of *stern*, yielding two possible candidates.

3.3.3 Synchronous grammar, transducers, and CCG

At this stage, let us return briefly to our discussion of the differences among the tree transformation- and CCG-based approaches to semantic parsing reviewed in Chapter 2. First of all, we note that the "tree transformation" approach could actually be further broken down into approaches that employ tree transducers and those that em-

ploy synchronous grammar to map between trees, two different formalisms that were developed by separate communities with a close relationship, so we could, in fact, view the two popular semantic parsing paradigms as three. However, all three can be viewed as simply different ways of defining a function mapping from items in one language (the language of meaning representations, in our case) to another (natural language). Under the tree transducer view, typically one conceptualizes the function by describing an automaton that walks a tree-shaped meaning representation and translates it step by step into a sentence, while under the synchronous grammar paradigm the function is defined by derivations which simultaneously generate the meaning representations and sentences. CCG takes a third view by defining a homomorphism between syntax and semantics.

In fact, all three not only perform the same task; but there are also deeper links in how they are often applied. Shieber (2004; 2014) showed that the tree transducers and synchronous grammars most commonly employed in NLP are equivalent, and even if one ventures into the realm of so-called "mildly context sensitive" grammars such as with synchronous Tree Adjoining Grammar, tree transducers and synchronous grammar still fall within a single overarching class of formalism. CCG in its most general form, has a somewhat less well studied relationship to the other two formalisms, but in computational settings, practitioners often restrict themselves to a simplified form which amounts to a variety of CFG, placing it in the domain of synchronous context-free grammar. Thus, no matter the paradigm, the vast majority (if not all) approaches to semantic parsing can be identified as not only working on the same problem, but also using formalisms of the same class with the same basic expressive power.

There are various different formalisms with varying expressive powers and computational complexities referred to as tree transducers, synchronous grammar, or CCG. For the following discussion we will primarily restrict consideration to those restricted forms that have typically found their way into recent semantic parsing applications. In particular, for practical considerations, all three are typically restricted to context-free languages.

Considering this restriction, all three classes are equally expressive from the perspective of formal language theory, at least for the purpose of developing practical computational models, leaving it up to other considerations when one chooses which one to adopt for a particular purpose. For instance, CCG is supported by a well-developed linguistic theory and elegantly simulates various subtle linguistic phenomena, while synchronous grammar and tree transducers were developed largely by the-

oretical computer scientists with little consideration for linguistic theory, albeit with some early exceptions involving Transformational Grammar (Chomsky, 1957; Cooper, 1975; Rounds, 1970). However, the linguistic theory that lends CCG its elegance as a tool for modeling the syntax-semantics interface is less helpful for explaining the relationship between word meaning and other aspects of cognition such as perception, which deals heavily with phenomena that are usually viewed as lying outside the scope of linguistic theory. Thus, CCG would need to be adapted to the word learning task, requiring one either extend or compromise the linguistic theory. Synchronous grammars and transducers, on the other hand, are agnostic to any linguistic theory, simplifying the modeler's task to some extent. In particular, while it is common to think of synchronous grammar as generating a set of pairs (e.g., meaning representation-sentence pairs), it is easy to add a third dimension to any synchronous grammar so that it instead generates triples. Thus, it is relatively straightforward to extend a synchronous PCFG-based semantic parsing model to define a joint probability distribution over world-meaning-sentence triples, provided that the world is describable by some context-free formalism such as a HRG.

Finally, one also might consider computational complexity. Even in cases where CCG can be simulated by a CFG, the simulating grammar typically contains far more nonterminals than a typical CFG-based syntactic grammar, leading to increased parsing time. For this reason, among others, the existing transformation-based semantic parsing systems often require less computational resources. As a practical concern, this last fact is critical for our word learning experiments, where there is far more ambiguity with its accompanying computational complexity to contend with than in traditional semantic parsing.

Synchronous grammar can essentially be thought of as a generalization of tree transducers, with tree transducers as a restricted case dealing with regular tree languages, so the first choice is between the CCG and synchronous grammar/tree transducer classes. The two points of computational complexity and the ease of extending a model to the three-way relationship between world, meaning, and sentence leads us to opt for a synchronous grammar or tree transducer for our work. Furthermore, because there are important differences between the trees we work with and those of the typical regular tree language-based tree transducers, we will generally adopt a synchronous grammar view. As an additional benefit, it is also easier to link to the HRG literature with a synchronous grammar, helping to ground our work in a broader literature.

3.4 Multi-weighted probabilistic context-free grammar

By associating a non-negative real valued weight with each rule and enforcing certain simple constraints (specifically, that each weight is less than one, and that the weights of rules with the same left-hand side sum to one) allows one to leverage the power of grammars for defining probabilistic models over derivations and their yields, be they strings, graphs, trees, or, in the case of synchronous grammar, some combination of these. In particular, one can compute the probability of a derivation by simply multiplying the weights of the individual rules and to compute the probability of the yield by summing over all possible derivation trees. Such probabilistic grammars work best when one is interested in the full joint probability of the derivation, but are less well suited if one is only interested some marginal probability. In the grammar-based word learning models of Johnson et al. (2012; 2010), for instance, the grammars define a joint probability over the scene, the meaning, and the utterance, but the word learning objective is the lexicon itself consisting of isolated words paired with their meanings. Thus, evaluation requires integrating out many extra variables that, while necessary for modeling the full joint probability, are extraneous to the lexicon.

Computing the necessary marginals may be more or less difficult depending on the specific grammar, where independence assumptions inherent to the context-free nature of the derivations can be exploited to improve efficiency. However, it may be hard to model certain independence assumptions without drastically re-factoring the grammar which can have negative side effects, such as changing the class of grammar rendering certain class-specific algorithms, such as for parsing, inapplicable. This problem may be particularly severe in the case of synchronous grammar since the amount of information in a synchronous grammar rule is much larger than that of the corresponding rules in monolingual grammars, meaning that there is more to integrate out, and, even worse, synchronous grammars are often harder or even impossible to re-factor into smaller rules without changing the language of the grammar. In these cases it may be necessary to work with a less desirable model parameterization and then compute rule-internal marginals, such as, for example, the conditional probability of the utterance given its meaning representation, by renormalizing rules in a post-processing step. However, such a post-processing step does not solve the problem of how to enforce independence assumptions within the model when they cannot be modeled by a simple re-factoring of the grammar, something that will prove necessary for implementing the models in Chapters 6 and 8.

This section introduces a novel generalization of conventional probabilistic grammars that addresses some of these problems without the need for either re-factoring the grammar or for a post-processing step by re-interpreting each rule as a collection of multinomial random variables and computing the rule weight as a product of their individual probabilities. The dependencies between these rule-internal variables can be manually specified, allowing one to factorize the parameters of the model without necessarily re-factoring the grammar rules themselves. Multiplying these rule-internal probabilities together results in a conventional weighted grammar, but the factorization additionally defines a rule-internal Bayesian network over the various features that make the rule. In effect, while in a conventional PCFG, the rule right-hand-side is generated all at once as a single monolithic event, in a multiweighted grammar the right-hand-side is generated from several smaller events governed by a Bayes net. These local Bayesian networks can define rule-internal independence assumptions, expanding the space of models one can express without necessarily changing the grammar rules themselves.

For example, the following two rules can be thought of as describing two different probabilistic outcomes, each specifying how nonterminal A is expanded in a derivation.

$$\mathrm{A} \rightarrow \mathrm{B\ C}$$
$$\mathrm{A} \rightarrow \mathrm{D\ C}$$

In a conventional PCFG, there are two random variables and a single probabilistic dependency between them, one corresponding to the conditioning information (i.e., the left-hand side symbol), and one corresponding to the expansion of the left-hand side symbol as follows:

$$\underbrace{\mathrm{A}}_{lhs} \rightarrow \underbrace{\mathrm{B\ C}}_{rhs}.$$

The weights on the two example rules together help define the probability distribution $P(rhs|lhs = \mathrm{A})$, where, in one case, the string B C is sampled for *rhs*, and in the other D C is sampled.

However, under the multi-weighted generalization, the event drawn for *rhs* can be further broken down into smaller sub-events, leading to a further factorization of the probability of A's expansion. The outcome for *rhs* can, say, be broken down into two separate variables rhs_0 and rhs_1 corresponding to the first and second symbols of the

string on the right-hand side as illustrated below:

$$\underbrace{\text{A}}_{lhs} \rightarrow \underbrace{\text{B}}_{rhs_0} \underbrace{\text{C}}_{rhs_1}.$$

If we like, we can choose to model rhs_0 and rhs_1 independently of one another given *lhs*, leading to the following factorization:

$$P(rhs = \text{B C} | lhs = \text{A}) = P(rhs_0 = \text{B} | lhs = \text{A}) \cdot P(rhs_1 = \text{C} | lhs = \text{A}).$$

There are many additional ways of factorizing $P(rhs|lhs)$, either using variables rhs_0 and rhs_1 or some other way of breaking down the rules.

These multi-weighted grammars are related to the locally-normalized local feature models described by (Berg-Kirkpatrick et al., 2010), which, when applied to a PCFG-based model, would weight rules according to a logistic regression parameterized by features of the rule. That is, like our multi-weighted grammars, each rule is identified with its own local probabilistic model, but in our case the model is a Bayesian network of multinomial variables instead of a logistical regression model. One advantage of a Bayesian network over logistical regression is that a Bayesian network permits one to manually specify conditional independence assumptions among particular features. As a consequence, it requires extra work to compute marginal probabilities of specific features in logistic regression, while, in a Bayesian network, these may be built into the parameterization of the model, depending on the specific network. Another advantage is that since the variables of the Bayesian networks are restricted to be multinomial, the result is a product of multinomials just like for ordinary PCFGs, allowing one to employ essentially the same class of inference and estimation algorithms (e.g., maximum likelihood).

The cascades of probabilistic grammars and automata often employed in machine translation to factorize larger problems into smaller sub-problems are also of a similar spirit, and can be seen as a special case of multi-weighted grammar. For instance, Knight and Graehl (1998) describe an approach to modeling English loan words in Japanese as a three stage process involving (1) mapping a word in English to a phonemic representation, (2) applying Japanese phonological rules to adapt this English phonemic representation to a Japanese phonemic form, and (3) mapping the Japanese phonemes to the Japanese script. While each stage is modeled by a relatively simple string automaton with its own weights, they can be composed into a single automaton that models the full joint probability of the entire cascade all at once, where the

weights of the composed automaton are computed by multiplying those of the individual sub-automata. Chiang et al. (2010) describe an algorithm for estimating the weights of each individual automaton using only the source and target surface forms by essentially training the composed machine while tracking the individual factors to each composite rule weight contributed by the sub-machines. In fact, the composed automaton is effectively a multi-weighted grammar where the Bayesian network associated with each rule describes how the weights of the individual sub-automata combine, and our inference algorithm described in Chapter 5 is essentially a generalization of the Chiang et al. (2010) training algorithm.

However, multi-weighted grammars are more general than composed probabilistic automata, since a multi-weighted grammar need not be the product of composing or intersecting separate grammars. Indeed, there is no general composition algorithm for HRG, since hyperedge replacement languages are not closed under intersection. However, one can still define a factorization of rule weights that produce the same effect as one might wish to model with such a cascade of simpler grammars.

Formally, each rule is broken down into a sequence of *factors* by a *rule factorization function* $\varphi : \mathcal{R} \to (\mathcal{X} \times \mathcal{X})*$, where $\mathcal{X}$ is the set of possible rule features. That is $\varphi(r) = \varphi_1(r) \cdot \varphi_2(r) \cdot \ldots \cdot \varphi_n(r)$, essentially returns a Bayes net for rule r where each $\varphi_i(r)$ is a pair of features of rule r of the form $\langle c, e \rangle$ defining the directed edges of the network. Each of these feature pairs is assigned a weight so that there is also effectively a *weight function* $\omega : \mathcal{R} \to \mathbb{R}^*$, which maps each rule r to a weight factorized into a sequence of real numbers, $\omega_1(r) \cdot \omega_2(r) \cdot \ldots \cdot \omega_n(r)$, where $\omega_1(r)$ corresponds to $\varphi_1(r)$, $\omega_2(r)$ to $\varphi_2(r)$, and so on. Thus, $\varphi_i(r) = \langle e, c \rangle$ corresponds to probability $P(e|c) = \omega_i(r)$, and, when multiplied out, $\omega(r)$ defines the full weight of the rule $\prod_{i=1}^{n} \omega_i(r)$.

In the preceding example with rule A $\to$ B C, *rhs*, rhs_0, and rhs_1 are all essentially feature functions, where, in the ordinary PCFG case,

$$\varphi(r) = \langle lhs(r), rhs(r) \rangle$$
$$\omega(r) = P(rhs(r)|lhs(r))$$

and in the second case

$$\varphi(r) = \varphi_1(r)\varphi_2(r) = \langle lhs(r), rhs_0(r) \rangle \langle lhs(r), rhs_1(r) \rangle$$
$$\omega(r) = \omega_1(r)\omega_2(r) = P(rhs_0(r)|lhs(r))P(rhs_1(r)|lhs(r)).$$

Figure 3.4 illustrates a few examples of different possible features; they could iden-

tify substrings of rule right-hand sides, a count of the nonterminals, or some combination of these and other features. As in the PCFG case where weights for rules fall in the range 0 to 1, each $\omega_i(r)$ must also fall between 0 and 1. Likewise, where in the PCFG case weights for rules with the same left-hand side sum to one, in this more general setting we have the constraint that all weights corresponding to feature pairs with the same conditioning information c must sum to one. That is, $\sum_{r,i:\varphi_i(r)=\langle c=k,e\rangle} \omega_i(r) = 1$ for each fixed feature value k.

One can view a parse under a multi-weighted grammar as a single Bayesian network where each rule supplies some module which are bound together at the nonterminals of rules. Each of these modules can be defined fairly flexibly, but there are some constraints. To maintain consistency with conventional probabilistic grammars, ω can only assign weights to features of the right-hand side, which are typically conditioned on, but do not include, the nonterminal on the left-hand side (i.e., there can be no cycles in the Bayes net). This way, derivation probabilities can be computed in the conventional way by simply multiplying rule weights together without fear of multiplying the probability of each nonterminal more than once. Similarly, nonterminals on the right-hand side should be accounted for by the probabilities of the Bayesian network. Otherwise, the derivation weight will fail to define a full joint probability of the variables of the network.

Let us consider some examples. Figure 3.4 contains two rules, $r2.3$ and $r3.3$, where several possible features and their integer identifiers have been indicated. In the HRG rule $r2.3$, feature function f_1 corresponds to the left-hand side and (VP in this case) and f_2 to the external vertices and terminal root edge (*scold*) of the graph on the right-hand side. To construct a conventional probabilistic grammar, we would define $\varphi(r2.3)$ to be just a single pair $\langle f_1(r2.3), f_5(r2.3)\rangle$, where f_1 identifies the left-hand side and f_5 the entire rule, as shown in the figure. Assuming this same $\langle$left-hand side, rule$\rangle$ scheme were applied consistently over the grammar, it would result in a rule weight of

$$\omega(r2.3) = \omega_1(r2.3) = P(r2.3|\mathrm{VP}). \tag{3.3}$$

With the grammar in Figure 3.2, one would find that $f_1(r2.2) = f_1(r2.3)$, indicating that $\omega_1(r2.2)$ and $\omega_1(r2.3)$ must sum to one, forming a multinomial vector.

However, it is possible to define several alternative probabilistic models with the same underlying grammar by just changing the rule factorization with a redefinition of ω and φ. In particular, one might want to further factorize the probability of $r2.3$ so that the ADV and NP are independent. One could, for instance, construct a probabilistic

$$\overbrace{\underbrace{\overbrace{\text{VP}}^{f_1} \rightarrow \overbrace{\lambda e,d.\text{scold}(e)}^{f_2} \wedge \overbrace{\text{ADV}(e)}^{f_3} \wedge \overbrace{\text{NP}(e,d)}^{f_4}}_{f_5}}^{f_6=f_2\wedge\text{child count}} \tag{r2.3}$$

$$\overbrace{\underbrace{\overbrace{\text{VP}}^{f_1} \rightarrow \langle\, \overbrace{\underbrace{\lambda e,d.\text{scold}(e)}_{f_3,f_4,f_5} \wedge \underbrace{\text{ADV}_{\boxed{0}}(e)}_{f_4} \wedge \underbrace{\text{NP}_{\boxed{1}}(e,d)}_{f_5}}^{f_2} \;||\; \overbrace{\underbrace{\text{ADV}_{\boxed{0}}}_{f_6}\ \underbrace{\text{scolded}}_{f_7}\ \underbrace{\text{NP}_{\boxed{1}}}_{f_8}}^{f_{10}} \,\rangle}_{f_{11}=f_3\wedge\text{child count}}}^{f_9} \tag{r3.3}$$

Figure 3.4: HRG and synchronous HRG-CFG rules with example features for implementing several alternative multi-weighted models with the same grammar. In rule $r2.3$ feature function f_1 identifies the nonterminal of the left-hand-side, and f_2 identifies the entire right-hand-size graph, while f_6 includes f_2 as well as the number of nonterminals. Similarly, f_3 and f_4 correspond to those nonterminal edges, and f_5 identifies the entire rule itself. The features of Rule $r3.3$ are defined in a similar manner, where, for example, feature functions f_4 and f_5 indicate particular subgraphs (excluding NP and ADV, respectively).

model that first selects the predicate and the number of children the predicate has, and then proceeds to choose the particular nonterminals for those children one at a time. This scheme could be implemented as follows:

$$P(r2.3|\text{VP}) = \omega(r2.3) = \omega_1(r2.3) \cdot \omega_2(r2.3) \cdot \omega_3(r2.3) \tag{3.4}$$

where we have chosen $\varphi(r2.3) = \varphi_1(r2.3) \cdot \varphi_2(r2.3) \cdot \varphi_3(r2.3)$ so that

$$\begin{aligned}
&\varphi_1(r2.3) = \langle f_1(r2.3), f_6(r2.3)\rangle \implies \\
&\qquad \omega_1(r2.3) = P(\,f_6(r2.3)|\,f_1(r2.3)) = P(\lambda e,d.\text{scold}(e), 2\text{ children}|\text{VP}) \\
&\varphi_2(r2.3) = \langle f_2(r2.3), f_3(r2.3)\rangle \implies \\
&\qquad \omega_2(r2.3) = P(\,f_3(r2.3)|\,f_2(r2.3)) = P(\text{ADV}(e)|\lambda e,d.\text{scold}(e)) \\
&\varphi_3(r2.3) = \langle f_2(r2.3), f_4(r2.3)\rangle \implies \\
&\qquad \omega_3(r2.3) = P(\,f_4(r2.3)|\,f_2(r2.3)) = P(\text{NP}(e,d)|\lambda e,d.\text{scold}(e))
\end{aligned}$$

In this case, $\omega_1(r2.3)$ is the probability of generating a *scold* predicate with two children ($f_6(r2.3) = \lambda e, d.\text{scold}(e) \wedge e\text{has 2 children}$) given a VP on the left-hand side ($f_1(r2.3) = \text{VP}$). The probability of ADV appearing as a child given parent predicate *scold* is indicated by $\varphi_2(r2.3) = \langle f_2(r2.3), f_3(r2.3)\rangle$, and the NP is generated in exactly the same manner according to $\varphi_3(r2.3)$. In this way, a multi-weighted grammar can implement a greater range of models than could an ordinary probabilistic HRG such as the one in Example 3.3 where the predicate and the child nonterminals are all generated in a single step without any independence assumptions.

The functions ω and φ behave exactly the same if the underlying grammar is synchronous. To define a conventional probabilistic synchronous grammar that jointly generates both the meaning and sentence using rules such as $r3.3$ in Figure 3.4, we would define φ to be a scalar as follows:

$$\varphi(r3.3) = \langle f_1(r3.3), f_9(r3.3)\rangle \implies \omega_1(r3.3) = P(r3.3|\text{VP}).$$

That is, just as with the conventional probabilistic HRG in Example 3.3, we assign one weight per rule and condition on the left-hand side.

Weighted synchronous grammars are also often used to implement conditional distributions, like, for instance, the probability of generating a sentence given a particular meaning representation. This scheme can again be implemented with a single weight per rule, but where the rule probabilities are conditioned on the meaning portion of the rule as well as the nonterminal on the left-hand side, giving us a $\varphi(r3.3) = \varphi_1(r3.3)$ of $\langle f_2(r3.3), f_9(r3.3)\rangle$, implying

$$\omega_1(r3.3) = P(r3.3|\lambda e, d.\text{scold}(e) \wedge \text{ADV}_{\boxed{0}}(e) \wedge \text{NP}_{\boxed{1}}(e, d), \text{VP}). \tag{3.5}$$

We can also model the effect of composing rules $r2.3$ and $r3.3$ so that the rule weights are the product of those of Examples 3.3 and 3.5, modeling the generation of the meaning representation followed by its translation into English.

$$P(r3.3|\text{VP}) = \omega_1(r3.3) \cdot \omega_2(r3.3) \tag{3.6}$$

Where φ is defined so that $\varphi(r3.3) = \langle f_1(r3.3), f_2(r3.3)\rangle$ and $\varphi(r3.3) = \langle f_2(r3.3), f_{10}(r3.3)\rangle$ to indicate conditioning information of the left-hand side and $\langle$left-hand side, meaning$\rangle$, respectively, yielding probabilities

$$\omega_1(r3.3) = P(\lambda e, d.\text{scold}(e) \wedge \text{ADV}(e) \wedge \text{NP}(e, d)|\text{VP})$$
$$\omega_2(r3.3) = P(\text{ADV}_{\boxed{0}}\ \text{scolded}\ \text{NP}_{\boxed{1}}|\lambda e, d.\text{scold}(e) \wedge \text{ADV}_{\boxed{0}}(e) \wedge \text{NP}_{\boxed{1}}(e, d), \text{VP}).$$

Finally, it is also possible to factorize synchronous rule probabilities just as we did for the HRG rule in example 3.4.

$$P(r3.3|\text{VP}) = \overbrace{\omega_1(r3.3)\cdot\omega_2(r3.3)\cdot\omega_3(r3.3)}^{P(\text{meaning}|\text{lhs})}\cdot\overbrace{\omega_4(r3.3)\cdot\omega_5(r3.3)\cdot\omega_6(r3.3)}^{P(\text{words}|\text{meaning})}$$

For $P(\text{meaning}|\text{lhs})$, i.e., the probability of the meaning given the left hand side, we assign the same factorization as for Example 3.4.

$$\begin{aligned}
\varphi_1(r3.3) &= \langle f_1(r3.3), f_{11}(r3.3)\rangle \implies \omega_1(r3.3) = P(\lambda e,d.\text{scold}(e), 2\ \textit{children}|\text{VP})\\
\varphi_2(r3.3) &= \langle f_2(r3.3), f_4(r3.3)\rangle \implies \omega_2(r3.3) = P(\text{ADV}(e)|\lambda e,d.\text{scold}(e))\\
\varphi_3(r3.3) &= \langle f_3(r3.3), f_5(r3.3)\rangle \implies \omega_3(r3.3) = P(\text{NP}(e)|\lambda e,d.\text{scold}(e))
\end{aligned}$$

For $P(\text{words}|\text{meaning})$, i.e., the probability of the sentence given the meaning, we assign a similar factorization, where we first generate the verb and the number of left and right arguments, and then choose the locations for the ADV and NP arguments conditioned on the meaning.

$$\begin{aligned}
&\varphi_4(r3.3) = \langle f_3(r3.3), f_7(r3.3)\rangle \implies\\
&\qquad \omega_4(r3.3) = P(\text{scolded}, 1\ \textit{left}, 1\ \textit{right}|\lambda e,d.\text{scold}(e))\\
&\varphi_5(r3.3) = \langle f_4(r3.3), f_6(r3.3)\rangle \implies\\
&\qquad \omega_5(r3.3) = P(\text{ADV}, \textit{left}|\lambda e,d.\text{scold}(e)\wedge\text{ADV}(e))\\
&\varphi_6(r3.3) = \langle f_5(r3.3), f_8(r3.3)\rangle \implies\\
&\qquad \omega_6(r3.3) = P(\text{NP}, \textit{right}|\lambda e,d.\text{scold}(e)\wedge\text{NP}(e,d))
\end{aligned}$$

Finally, in principle, we should also assign a probability to generating the bijection function $\frown$. This step was folded into the final step of Example 3.6 to simultaneously generate the bijection and the sentence, but it is also possible to include it as a separate step and weight it with yet another factor. However, if we constrain the grammar to only match nonterminals in the meaning with nonterminals of the same type in the syntax, the choice is deterministic (in the case of this rule), so one can omit this factor (which is a constant of 1) without impacting the probabilities.

3.4.1 Tied weights

It is also possible to define f so that the same factor is shared across multiple rules, which can be useful for tying parameters. For instance, consider the factorization

scheme of Example 3.6 where each rule r has two factors, $\omega_1(r)$ which models the probability of the meaning representation, and $\omega_2(r)$ which models the conditional probability of the utterance given the meaning. One might want rules $r3.3$ and $r3.9$ of the example grammar in Table 3.1 to have the same probability of generating meaning representations with and without *stern*. This parameter-tying could be accomplished by simply defining φ such that rules $r3.3$ and $r3.9$ share a feature pair $\varphi_1(r3.3) = \varphi_1(r3.9)$, resulting in a common factor $\omega_1(r3.3) = \omega_1(r3.9)$. This kind of parameter tying can be useful for smoothing purposes (Headden et al., 2009) or for integrating out certain details like in this case, where the model would effectively smooth over the decision of whether to introduce ADV-type modifiers such as *stern* in the meaning representation for *scold* events.

3.5 Conclusion

The formalisms laid out in this chapter establish a general framework for modeling a wide range of problems. Most of the ideas are already well established and thoroughly studied with their own set of algorithms, and we will make extensive use of this fact in the semantic parsing and word learning models described in Chapters 6 and 8. Even the more novel elements such as the synchronous grammars with mono-lingual rules in Section 3.3.1 or the multi-weighted grammars of Section 3.4 are closely tied to previous work with probabilistic context-free grammar with its extensive set of algorithms. For instance, the same general parsing algorithms used for CFG can be used for synchronous grammars with mono-lingual rules, and multi-weighted grammars are built on top of context-free style grammars and make use of most of the same algorithms for inference, such as probabilistic CKY.

This is not to say that innovation is completely unnecessary, but the novel components have a higher potential applicability due to their relationship to a common, general framework. Chapter 4 describes a novel parsing algorithm optimized for working with a particular subclass of HRG for describing languages of unordered tree, and Chapter 5 lays out the derivation of an inference algorithm for estimating the rule weights of multi-weighted grammars. Yet, even in these cases, the choice to work with formalisms so closely related to CFG and the family of similar formalisms such as regular tree grammar will prove useful. In fact, the unordered tree parsing algorithm is closely based on a standard algorithm for RTG parsing, and, as we will see, the training algorithm for multi-weighted grammars follows a form that may already be

familiar from a similar algorithm for PCFGs and makes use of other common tools such as the inside-outside algorithm for computing the expected counts of rules from a parse forest. Furthermore, by focusing on general formalisms rather than ad hoc solutions, it becomes much easier for others to make use of these algorithmic innovations, potentially applying them to very different problems.

Chapter 4

Parsing Unordered Trees

This chapter describes the parsing algorithm for unordered trees which we use for the word learning and semantic parsing models in Chapters 6 and 8. We make use of hyperedge replacement grammar for our parsing model, but restrict the formalism to a specialized class that only generates languages of unordered trees. The choice of context-free graph grammars as the basic underlying formalism makes it easy to integrate the grammars with context-free string grammars for semantic parsing while still modeling the commutative property of the predicate calculus of the meaning expressions. However, graph parsing is exponential in the most general case, and even the algorithm of Chiang et al. (2013) which is polynomial in the graph for our grammars is still quite expensive. Instead, we define a parsing algorithm which is general enough for a broad class of tree languages but sufficiently specialized to allow relatively efficient parsing. In fact, the grammars we work with are roughly analogous to regular tree languages where the trees are unordered, and, also like regular tree grammar, parsing is linear in the size of the tree. At a very high level, the parser essentially explores every possible permutation and parses as per a regular tree grammar while fixing each possible ordering. Of course, in spite of this theoretical linear time complexity bound, parsing can still be surprisingly expensive due to the fact that there is a large number of possible orderings, a number that is exponential in the grammar. Consequently, while the basic parsing algorithm itself is very simple and can be presented fairly quickly, the bulk of the chapter is dedicated to describing optimizations for reducing the combinatorial blowup in the grammar constant.

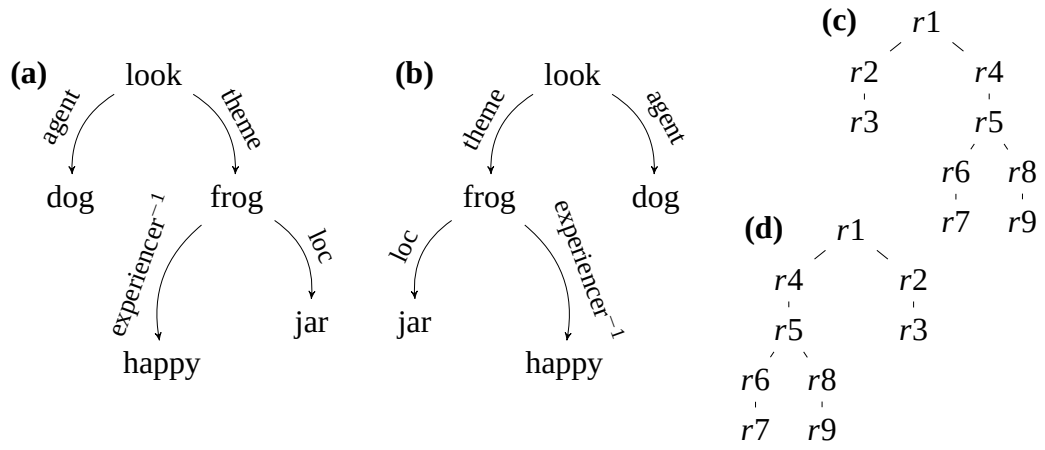

Figure 4.1: (a)-(b)Two equivalent unordered trees. (c)-(d) Two distinct derivation trees (which are ordered).

4.1 Unordered vs. ordered trees

Derivation trees such as (c) and (d) in Figure 4.1 are ordered, where the order dictates which nonterminal is expanded by which rule. We can see from the derivation tree in (c) that rule $r1$, for instance, has two nonterminals. The first nonterminal is expanded by applying rule $r2$ and the second by $r4$. This contrasts with (d), where the opposite is true, i.e., the first nonterminal is expanded by $r4$ and the second by $r2$. Because the two different orderings of $r2$ and $r4$ indicate different expansions for each of the nonterminals and therefore represent distinct sets of derivations, the ordering is important. This same convention of using the order of siblings in the derivation tree to indicate which nonterminals are expanded by which rules is general to context-free formalisms of all kinds, producing ranked, ordered derivation trees. In fact, one test for context-freeness of a particular formalism is whether its derivation trees form a regular tree language (sets of trees which are necessarily ranked and ordered).

However, trees used to represent predicate calculus expressions such as (a) in Figure 4.1 ideally should not be sensitive to order, since the logical interpretation is invariant over alternative orderings such as (b). That is, both trees are logically equivalent to the expression

$$\text{look}(e) \wedge \text{theme}(e,y) \wedge \text{frog}(y) \wedge \text{loc}(y,z) \wedge \text{jar}(z) \wedge \text{experiencer}^{-1}(y,s) \wedge \text{happy}(s) \\ \wedge \text{agent}(e,x) \wedge \text{dog}(x).$$

In fact, the reordering of the tree is similar to reordering conjuncts in the logical expression, where the commutative property of the conjunction operation guarantees that the interpretation does not change. Similarly, reordering of siblings in an unordered tree does not change its interpretation. Since ordering does not matter for the conjuncts in predicate calculus due to the commutative property, neither should the ordering of siblings in the corresponding tree representation. Thus, while ordered trees are necessary for representing parses, we use unordered trees for meaning representation expressions. Parts (a) and (b) of the figure can thus be thought of as merely different ways of drawing the same tree which corresponds to a single meaning representation expressed in predicate calculus notation.

To capture this unordered-ness property effectively one must use an appropriate formalism, where we choose a subset of HRG which generates languages of unordered trees. To understand the dichotomy between ordered vs. unordered formalisms, note that an ordered formalism would be forced to treat the two trees (a) and (b) in Figure 4.1 as distinct, potentially leading to different parses and different probabilities. Thus, under some particular grammar, parse (c) might yield tree (a) and (d) might yield (b). For instance, one may wish to use a grammar to define a language of meaning representations as defined by set of predicate calculus expressions. An ordered formalism would not be ideal for this given that it might both accept and reject a particular expression depending on the particular tree being parsed. Similarly, one might wish to compute the probability of a predicate calculus expression using a weighted grammar, and a weighted ordered grammar might produce different probabilities depending on the tree. The problems could grow even more complicated when one considers synchronous grammars. An unordered formalism such as HRG, on the other hand, would assign the exact same set of parses to both (a) and (b), sidestepping these issues.

The fact that derivation trees are ordered while the yield is unordered results in there being many possible derivation trees with the same yield that differ only in order and otherwise are completely symmetric, like (c) and (d) in Figure 4.1. Thus, a parse forest tends to consist of many nearly identical trees, a fact that plays a central role in the parsing algorithm described in Section 4.3. First, however, we formally define the particular class of HRG we use for our tree languages.

4.2 Unordered tree-generating HRG

Next, we define the restricted class of HRG for generating unordered trees, formally expressed as a tuple $\langle \Sigma, \mathcal{N}, \mathcal{E}, \mathrm{S}, \mathcal{R} \rangle$ where

- Σ is an alphabet of terminals,
- $\mathcal{N}$ is an alphabet of unary edge-generating nonterminals,
- $\mathcal{E}$ is an alphabet of binary edge-generating nonterminals,
- $\mathrm{S} \in \mathcal{N}$ is the start symbol, and
- $\mathcal{R}$ is a set of rules.

Rules come in two specific forms: those that generate vertex labels in the form of unary edges and those that generate binary edges. For the following discussion assume that $\mathrm{N} \in \mathcal{N}$, $\mathrm{E}, \mathrm{E}_1, ..., \mathrm{E}_N \in \mathcal{E}$, and $\mathrm{a}, \mathrm{e} \in \Sigma$. Then the two types of rule are:

- **Node-generating rules** of the form

$$\mathrm{N} \rightarrow \lambda x.\mathrm{a}(x) \wedge \mathrm{E}_1(x) \wedge ... \wedge \mathrm{E}_n(x) \tag{4.1}$$

  generate a node label (in the form of a unary edge) and introduce the child nonterminals $\mathrm{E}_1, ..., \mathrm{E}_N$ that generate the edges leading to its subtrees.

- **Edge-generating rules** of the form

$$\mathrm{E} \rightarrow \lambda x_0.\mathrm{e}(x_0, x_1) \wedge \mathrm{N}(x_1) \tag{4.2}$$

  generate a binary edge and introduce the nonterminal that generates the subtree rooted at the child vertex.

There are a few key properties of these rules. For one, there is always exactly one external vertex (i.e., one argument of the lambda expression) in each rule right hand side. This feature enforces the tree property so that there are no reentrancies in the generated graph. Also, each right hand side contains exactly one terminal and zero or more nonterminals drawn either from $\mathcal{E}$ if the rule is edge-generating or $\mathcal{N}$ if it is node-generating. Noting that S is a node-generating nonterminal, derivations begin with a node-generating rule and alternate between edge-generating and node-generating rules. These HRG-based ranked, unordered tree grammars are analogous to regular tree grammars without hidden states, except that the child nonterminals $\mathrm{E}_1, ..., \mathrm{E}_n$ are unordered and the binary edges of the tree are labeled.

Algorithm 2 The basic parsing algorithm for unordered trees.

function PARSE(vertex-rooted tree $t(x)$)
  $Ch =$ parse chart for $t(x)$
  for $d = depth(t(x)), ..., 0$ **do**
    for all vertices y in $t(x)$ at depth d **do**
      $Ch[t(y)] \leftarrow$ MATCH_VERTEX$(Ch, t(y))$
      if $d > 0$ **then**
        $Ch[t(parent(y), y)] \leftarrow$ MATCH_EDGE$(Ch, t(parent(y), y))$
  return Ch

Algorithm 3 The vertex-matching algorithm. Returns all rule matches for the root vertex of tree $t(x)$ assuming all matches for the child edges of x have already been found.

function MATCH_VERTEX(parse chart Ch, tree $t(u) = \mathrm{a}(u) \wedge t(u, v_1) \wedge ... \wedge t(u, v_n)$)
  $Items \leftarrow \emptyset$
  for all rules $r = \mathrm{N} \rightarrow \lambda x.\mathrm{a}(x) \wedge \mathrm{E}_1(x) \wedge ... \wedge \mathrm{E}_n(x) \in \mathcal{R}$ **do**
    for all $\langle \mathrm{E}_{i_1}, ..., \mathrm{E}_{i_n} \rangle \in permute(\mathrm{E}_1, ..., \mathrm{E}_n)$ **do**
      if $\langle \mathrm{E}_{i_1} \rightarrow \sigma_1, \pi_1 \rangle \in Ch[t(u, v_1)] \wedge ... \wedge \langle \mathrm{E}_{i_n} \rightarrow \sigma_n, \pi_n \rangle \in Ch[t(u, v_n)]$ **then**
        $Items \leftarrow Items \cup \{\langle r, \langle i_1, ..., i_n \rangle\}\rangle\}$
  return $Items$

4.3 Frontier-to-root parsing

The parsing algorithm is similar to a regular tree grammar parser. Proceeding from the deepest nodes at the frontier of the tree and gradually working up the tree to the root, it first matches each node of the tree with a rule of type (4.1) and then matches its parent edge with a rule of type (4.2). Algorithm 2 presents the pseudocode. The main point where the algorithm deviates from a regular tree parser is in how it handles the lack of ordering among child edges on the right hand side of node-generating rules.

Before going into the details, let us first define some notation to denote subtrees of the yield, where there are two types of subtrees, vertex-rooted and edge-rooted subtrees. The subtree in Figure 4.1(a) rooted at the *frog* vertex corresponding to the logical expression $frog(x) \wedge experiencer^{-1}(x, s) \wedge happy(s) \wedge loc(x, y) \wedge jar(y)$ is an example of a vertex-rooted tree, and if we prepend the the binary edge for $theme(e, x)$ we end up with an edge-rooted tree. We will use $t(v)$ to denote the subtree rooted at vertex v, and $t(u, v)$ to denote $t(v)$ plus the edge linking v to its parent vertex u. We can define them

Algorithm 4 The edge-matching algorithm. Returns all rule matches for the root edge of tree $t(u,v)$ assuming all matches for vertex v have already been found.

function MATCH_EDGE(parse chart *Ch*, edge-rooted tree $t(u,v) = e(u,v) \wedge t(v)$)
  $Items \leftarrow \emptyset$
  for all rules $r = \mathrm{E} \rightarrow \lambda x_0.e(x_0,x_1) \wedge \mathrm{N}(x_1) \in \mathcal{R}$ **do**
    if there is a match $\langle \mathrm{N} \rightarrow \sigma, \pi \rangle \in Ch[t(v)]$ **then**
      $Items \leftarrow Items \cup \{\langle r, \langle 1 \rangle \rangle\}$
  return *Items*

recursively:

$$t(u) = a(u) \wedge t(u,v_1) \wedge \ldots \wedge t(u,v_n)$$

where $v_1, \ldots, v_n$ are u's children, and

$$t(u,v) = e(u,v) \wedge t(v).$$

As Algorithm 2 proceeds from the leaves toward the root, it visits each node- and edge-rooted tree finding parse items for node-generating and edge-generating rules. Parse items consist of a rule and a mapping from the nonterminals on its right-hand side to the child subtrees of the corresponding edge or node, which we denote by

$$\langle \mathrm{A} \rightarrow \beta, \pi \rangle$$

where π is a bijection between nonterminals in rule $\mathrm{A} \rightarrow \beta$ and the subtrees of the tree being parsed. In terms of the derivation tree, successfully finding such a match means that there is at least one derivation that expands each nonterminal in β to the subtree it maps to in *pi*.

These parse items are recorded in a parse chart which simply lists the set of parse items found for each subtree. The parse chart is a tree-shaped data structure in its own right where each node is identified with some node- or edge-rooted subtree of the ground term being parsed and consists of the set of parse items found for the corresponding root node or edge. As the parser visits each node and edge, the algorithm alternates between Algorithm 3 which enumerates parse items corresponding to the node-generating rules and Algorithm 4 which enumerates items for the edge-generating rules.

Note that each parse is just a tree of the same general shape as the tree being parsed (modulo sibling ordering considerations). Thus, all parses follow the same

shape and are in effect identical except for different labelings to indicate the different rules and permutations which are used. Consequently, the packed forest itself can also be represented as a single tree where each node is a composite structure containing all the different possible rule-permutation pairs for the subtree of the yield in the full set of parses.

Algorithm 4 describes the edge-matching procedure which essentially just looks up the set of rules that contain an instance of the binary edge to be matched which have nonterminals that are consistent with the previously parsed portion of the yield. Because edge-generating rules are comparatively simple, containing exactly one terminal edge and one nonterminal in parent-child relation, there are no ordering issues, and the parser behaves no differently from how one might proceed in parsing a fully ordered tree.

Algorithm 3 returns parse items for the node-generating rules of type (4.1) and is considerably more complex due to the need to account for the lack of ordering on the yield tree. In ordinary RTG or CFG parsing, the built-in ordering of the yield constrains which nonterminals can cover which subtrees or spans of the yield. As an example, suppose we had a CFG rule such as

$$\mathrm{S} \rightarrow \mathrm{NP\ VP}.$$

Due to the ordering constraint built into CFG, if we had matched the NP to a span of the yield string covering, for example, words 5 through 6, the VP could only be matched to a span starting at 6. However, this is not the case for an unordered tree grammar rule such as

$$\mathrm{EVENT} \rightarrow \lambda x.\mathrm{look}(x) \wedge \mathrm{ROLE}_1(x) \wedge \mathrm{ROLE}_2(x),$$

where the ROLE_1 nonterminal could be assigned to the *agent* and ROLE_2 to the *theme* of the tree in Figure 4.1(a) or vice versa. In fact, because the yield is unordered, any nonterminal on the right hand side of the rule can, depending on the specific grammar, generate any subtree of the yield. Thus, instead of simply matching each nonterminal to a subtree according to a set order as one might do if the yield were ordered, we must explore all the possible mappings between nonterminals and subtrees. Each such mapping between nonterminals and subtrees corresponds to a particular permutation of the nonterminals on the rule right hand side (or, equivalently, subtrees in the yield). Thus, much of the work of the parser consists of enumerating permutations, and parse items are constructed by temporarily enforcing each of these possible orderings of the nonterminals and parsing as though the tree were fully ordered.

Treating the grammar as constant, the complexity is exactly the same as for parsing in the ordered tree setting with regular tree grammars. That is, assuming a tree consisting of $|m|$ vertices, the algorithm has both $O(|m|)$ time and memory complexity. The complexity of both the edge- and node-matching algorithms are absorbed into the grammar constant, since the number of rules and permutations that must be explored are entirely grammar dependent. Nevertheless, the grammar constant can be quite large, especially if there are large grammar rules with many nonterminals on the right hand side (resulting in many possible permutations that must be explored during vertex-matching). In fact, our problem is closely related to that of Immediate Dominance/Linear Precedence parsing (Shieber, 1984), a generalization of General Phrase Structure Grammar parsing intended to handle properties of so-called free-word-order languages. Barton (1985) pointed out that the lack of ordering on rule right-hand sides in these grammars leads to time complexity that, while polynomial in the string, is exponential in the size of the rule right-hand sides. We are faced with the same problem, as is any more general HRG parser such as Chiang et al. (2013). However, there is a relatively efficient method of encoding many of the necessary permutations which can help bring the grammar constant down to something more practical, an optimization described in the next sections.

4.3.1 A compact encoding for symmetric parses

It is possible for distinct, but symmetric, parses to generate isomorphic trees. In fact, there can be an exponential number of such symmetric parses. However, it is possible to detect such symmetries even before explicitly enumerating them by simply looking at the rules of the grammar, suggesting an optimization that can greatly improve parsing performance both in terms of memory and time complexity. Under this optimization, instead of exhaustively enumerating all the symmetric parses, the parser need merely produce one from which the others can be constructed as necessary.

For instance, consider derivation trees in (c) and (d) of Figure 4.1. Assuming a

grammar like the following, both derivation trees generate the same unordered tree.

$$\text{EVENT} \to \lambda e.\text{look}(e) \wedge \text{ROLE}(e) \wedge \text{ROLE}(e) \quad \text{(r1)}$$

$$\text{ROLE} \to \lambda e.\text{agent}(e,x) \wedge \text{ENTITY}(x) \quad \text{(r2)}$$

$$\text{ROLE} \to \lambda e.\text{theme}(e,x) \wedge \text{ENTITY}(x) \quad \text{(r3)}$$

$$\text{ENTITY} \to \lambda x.\text{dog}(x) \quad \text{(r4)}$$

$$\text{ENTITY} \to \lambda x.\text{frog}(x) \wedge \text{ROLE}(x) \wedge \text{ROLE}(x) \quad \text{(r5)}$$

$$\text{ROLE} \to \lambda x.\text{experiencer}^{-1}(x,e) \wedge \text{AFFECT}(e) \quad \text{(r6)}$$

$$\text{AFFECT} \to \lambda e.\text{happy}(e) \quad \text{(r7)}$$

$$\text{ROLE} \to \lambda x.\text{loc}(x,y) \wedge \text{PLACE}(y) \quad \text{(r8)}$$

$$\text{PLACE} \to \lambda x.\text{jar}(x) \quad \text{(r9)}$$

Although the derivation trees are distinct, they are perfectly symmetric, include the exact same rules, and have the same probability no matter what weights are assigned to the rules of the grammar. These permutations arise from the internal symmetries within the branching rules $r1$ and $r5$. Each of the two rules has two possible identical permutations, which combine in a multiplicative fashion to form a total of four possible parses, of which (c) and (d) are just two. For probabilistic purposes we need not list all four; instead we can annotate the parse items corresponding to the applications of $r2$ and $r5$ with the number of symmetries. These extra symmetry counts can be ignored if one simply wants the probability of a single tree, but if one wants to, say, compute the sum of all trees such as when calculating inside probabilities in the inside-outside algorithm, one can simply multiply them into the product of the rule weights. For instance, when computing the inside probability of node $r5$, we multiply the probabilities of the sub trees rooted at $r6$ and $r8$ by that of $r5$ as well as by 2, the number of symmetries for the rule. Similarly, applying the same procedure for $r1$ multiplies in an additional factor of 2, so the total probability of the parse and its 3 other symmetric brethren is just the probability of one of these parses times $4 = 2 \cdot 2$.

More generally, consider node-generating rules of type (4.1). There may be multiple nonterminal edges sharing the same label so there are only $k \leq n$ distinct symbol types among the $E_1, \ldots, E_n$. Let these symbols be $\{A_1, \ldots, A_k\}$ where there are n_i instances of each A_i among the $E_1, \ldots, E_n$. Then there are $\prod_{i=1}^{k} n_i!$ permutations of the tree on the right hand side of the rule that are all identical to one another. This implies that if a single match is found for rule r, there may be as many as $\prod_{i=1}^{k} n_i!$ symmetric

matches[1]. This is because, given the lack of ordering on the yield, if edge $E_i(x)$ can be expanded to produce a particular subtree of the yield, so too could any other nonterminal edge $E_j(x)$ of the same label (i.e., where $E_j = E_i$), and $E_j(x)$ can be expanded according to the exact same set of derivations as could $E_i(x)$.

Because the symmetric parses all have the same probability and the same yield, for the purposes of finding the most probable parse or computing the expectation of a rule, they are all equivalent, making it sufficient to store just one instead of all $\prod_{i=1}^{k} n_i!$ parses. To preserve probability mass in the parse forest, we can annotate each parse item with the number of symmetries in the match, which we will denoted $syms_r$, and multiply this in when computing the total probability of the symmetric parse trees. That is, to compute the total probability of a particular parse plus all of if its symmetric brethren, one simply needs to compute a single product:

$$\prod_{r \in X} \omega_r \cdot syms_r, \tag{4.3}$$

where X is a specific parse, r is a particular instance of a grammar rule in X, ω_r is r's weight, and $syms_r$ is the total number of symmetries involving r, a quantity we show how to compute in equation 4.4.

The result is a reduction in memory usage by a factor that can be as large as $\prod_{i=1}^{k} n_i!$ for each node in the tree being parsed since there are this many fewer parse items that must be explicitly stored. Furthermore, the function *permute* in Algorithm 3 can be implemented to only return the distinct permutations, thereby potentially reducing time complexity as well (see Knuth (2005) for an algorithm for enumerating multiset permutations). The number of permutations that we must consider is still considerable, $\frac{n!}{\prod_i n_i!}$, but it may be a sizable reduction compared to $n!$.

4.3.2 Excluding duplicate parses

While we take care to count the symmetric parses even as we forgo explicitly enumerating them, in doing so it is important that we do not double count duplicate parses. The full set of $\prod_i n_i$ permutations will sometimes include identity permutations that produce multiple copies of the same parse tree, each of which should only be counted once. Otherwise algorithms like the inside-outside algorithm will over-estimate the total probability mass of the forest. This case comes up when a vertex in the yield tree has multiple identical subtrees among its immediate descendants. In matching a

[1] This $\prod_{i=1}^{k} n_i!$ term is an upper bound. We will return to this in Section 4.3.2

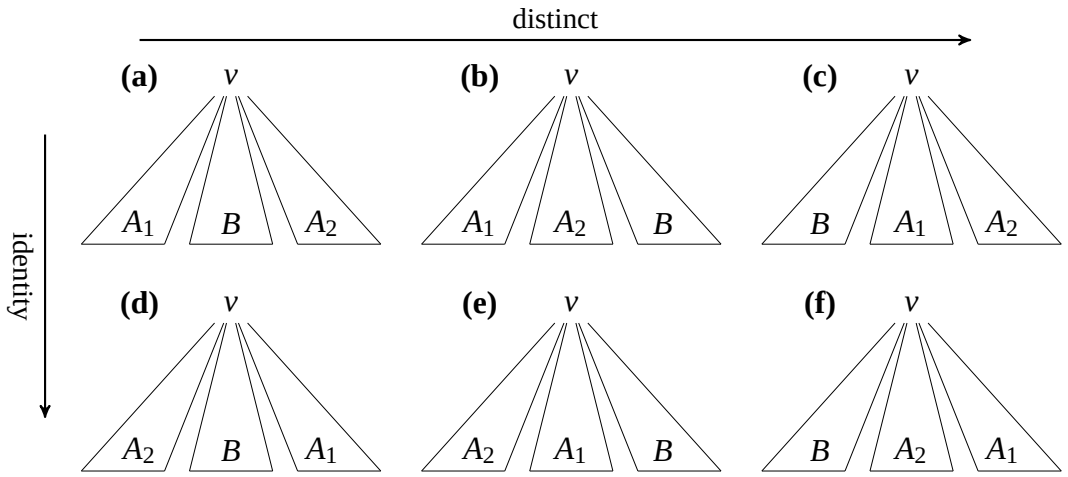

Figure 4.2: The six isomorphic permutations of an unordered tree with two identical subtrees. The permutations can be factorized into those producing distinct orderings (horizontal axis) and the identity permutations arising from the identical A_1 and A_2 subtrees (vertical axis).

rule with such a tree, a parser must account for all possible pairings between the nonterminals of the rule with the subtrees of the yield, effectively involving enumerating permutations of the tree. However, owing to the identical subtrees, some permutations lead to identical trees, resulting in duplicate parses.

Consider the example tree and its six permutations in Figure 4.2. Each of these permutations correspond to a particular rule matching during parsing, and thus a set of parses that include that particular rule matching. For example, given a rule such as

$$\mathrm{N} \rightarrow \lambda x.\mathrm{a}(x) \wedge \mathrm{C}(x) \wedge \mathrm{D}(x) \wedge \mathrm{E}(x),$$

(a) corresponds to an assignment of nonterminal C to subtree A_1, D to B, and E to A_2. The assignment corresponding to (d) would produce the same result, since subtrees A_1 and A_2 are identical and it does not matter whether they are respectively paired off with C and E or E and C. The same holds for permutations (b) and (e) as well as (c) and (f). Thus, there are only three distinct sets of parses rather than six, and a parser should only count the three, omitting the others as duplicates.

To prevent this kind of double counting of parses, the parser must filter out the permutations that lead to identical parses. Duplicate parses can only arise from the identity permutations, since a unique permutation of nonterminal symbols necessarily

produces a unique parse. Thus, we focus on those permutations that have been summarized by the symmetry counts for each parse used in the compact symmetry encoding scheme. Say, one were matching a rule with three identical nonterminals such as

$$\mathrm{N} \rightarrow \lambda x.\mathrm{a}(x) \wedge \mathrm{E}(x) \wedge \mathrm{E}(x) \wedge \mathrm{E}(x).$$

In this instance, the parser would produce a single parse item to cover all six cases since there is only one unique permutation for the rule, and would compute the maximum number of symmetries as $3! = 6$. However this total set of permutations can be factored into two different sets of permutations, the distinct and identity permutations, as illustrated in Figure 4.2, where each individual permutation can be reinterpreted as the composition of one of the distinct permutations (a), (b), or (c), followed by one of two identity permutations. In fact, the total set of permutations (6 in this example) is the product of the number of distinct permutations (3) and the number of identity permutations (2). Consequently, knowing the total number of permutations and the number of identity permutations allows one to compute the number of distinct symmetric parses.

In general, the number of identity permutations can be computed by counting the instances of each nonterminal-tree pair in the matching: 2 of $\langle \mathrm{E}, A \rangle$, and 1 of $\langle \mathrm{E}, B \rangle$ in this case. Say, there are ℓ such pair types and τ_i instances of each of these ℓ pairs. Then, the number of identity permutations is $\prod_{i=1}^{\ell} \tau_i!$. In the compact symmetry encoding scheme we compute the actual number of symmetries with which to annotate each parse item as

$$syms_r = \frac{\prod_{i=1}^{k} n_i!}{\prod_{i=1}^{\ell} \tau_i!}. \tag{4.4}$$

In our example this becomes $\frac{3!}{2!1!} = 3$. This $syms_r$ can then be used as per equation 4.3 to compute the total probability mass of a set of symmetric parses.

4.3.3 Canonical orders and hashing

As in CFG parsing, it is often useful to index rules by their right hand sides. For instance, one can optimize the vertex- and node-matching algorithms to simultaneously match all rules with isomorphic right hand sides. This is slightly more complicated in an unordered formalism since there are many isomorphic trees that may still be distinct because they differ in the order in which the children are listed in the rule

representation. For example, there could be two different rules:

$$\text{A} \rightarrow \lambda e.\text{lick}(e) \wedge \text{AGENT}(e) \wedge \text{THEME}(e)$$
$$\text{B} \rightarrow \lambda e.\text{lick}(e) \wedge \text{THEME}(e) \wedge \text{AGENT}(e)$$

One simple way of handling these orderings is to define a canonical ordering so that all isomorphic trees will receive the same representation, so the two rules in our example would have identical (not just isomorphic) right hand sides. A simple lexical sort on nonterminal labels suffices.

4.3.4 Correctness and Complexity

Determining each possible way a particular node-generating grammar rule could feature in a particular parse, one must consider all possible ways the nonterminals on the right-hand side might expand to produce the subtrees in the yield beneath the corresponding vertex. Each of these ways of generating the subtrees corresponds to a particular permutation of the nonterminals of the rule, matching the nonterminals to subtrees, implying that parses correspond to rule permutations. An exhaustive enumeration of such permutations covers all possible parses, but, as pointed out in Section 4.3.1, brute force enumeration is not only expensive but, in many cases, unnecessary. Employing the optimization described there, the parser avoids enumerating the unnecessary derivation trees by partitioning the parses into subsets of symmetric parses. For present purposes, we define the symmetry relation such that any pair of parses are symmetric if and only if either one can be constructed from the other by applying identity permutations to the grammar rules (i.e., permutations of the nonterminal symbols that result in the same sequence). The symmetry relation in turn defines a partition of the total set of possible parses, where any two parses appear within the same subset if and only if they are symmetric. We call these subsets the symmetry classes of the parse forest. Consequently, the parser need identify only one representative parse from each symmetry class since all other parses in the set can be reconstructed by permuting the parse tree according to the symmetries in the constituent grammar rules.

To show that this compact parse encoding scheme correctly accounts for all parses even without explicitly computing many of them, we must demonstrate that all nonterminal-to-subtree mappings are correctly accounted for without over- or under-counting any parses in the final parse forest. The full set of permutations for a rule with n nonterminals is simply $n!$, where, as discussed in Section 4.3.2, a certain proportion are

redundant since they lead to duplicate parses. In fact, taking into account these duplicates, there are actually only $\frac{n!}{\prod_{i=1}^{\ell}\tau_i!}$ permutations that lead to unique parses, where the $\tau_1,\ldots,\tau_\ell$ are the counts of the unique pairings between rule right-hand side nonterminal edge labels and their final yields according to the parse tree. The total number of permutations can be factored into the product of the multiset permutations (the permutations which lead to distinct label sequences) and the identity permutations (which result in identical label sequences):

$$\underbrace{\left(\frac{n!}{\prod_{i=1}^{\ell}\tau_i!}\right)}_{\text{total permutations}} = \underbrace{\left(\frac{n!}{\prod_{j=1}^{k}n_j!}\right)}_{\text{multiset permutations}} \cdot \underbrace{\left(\frac{\prod_{j=1}^{k}n_j!}{\prod_{i=1}^{\ell}\tau_i!}\right)}_{\text{identity permutations}},$$

where $n_1,\ldots,n_k$ are the number of occurrences of each of the unique nonterminal symbol labels among the n nonterminal edges.

First, consider only the explicitly enumerated permutations (the multiset permutations returned by *permute*) which we argue are in one-to-one relation with the symmetry classes. Multiset permutations only produce unique sequences of nonterminal symbols. However, according to our definition for "symmetric", symmetric parses can only be produced by permuting sibling subtrees of the parse with identical nonterminals at their roots. Consequently, given two distinct multiset permutations, it is impossible for one to be constructed from the other by applying any such permutation, implying that their corresponding parses must belong to separate symmetry classes. By a nearly identical argument, any two parses drawn from separate symmetry classes must also correspond to separate multiset permutations. Thus, there is a one to one relation between the multiset permutations of rule right-hand sides and the symmetry classes of the parses.

Then, to show that the algorithm neither over- nor under-counts parses, it suffices to demonstrate that the compact encoding scheme itself correctly accounts for all parses within a given symmetry class. That is, given a particular symmetry class and a representative parse, we must show a one-to-one relation between the identity permutations of the constituent rules and the symmetric parses themselves. By definition, given two parses from the same symmetry class one can be constructed from the other by permuting sibling subtrees that expand nonterminals of the same symbol. These permutations correspond to rule identity permutations. Thus, the only thing that remains to be shown is that there are no more identity permutations than there are parses in the symmetry class (i.e., they do not lead to duplicate parses), which we do by contradiction.

Assume that there are two distinct permutations that somehow lead to the same

parse tree. Since the permutations are distinct, we have a situation where at least two nonterminals map to separate subtrees of the yield. If the corresponding parses are identical, however, this requires that these two nonterminals, in spite of expanding to cover separate portions of the yield, still have identical expansions (otherwise the parses would differ). However, this situation is also addressed by Section 4.3.2, which eliminates this case from the symmetry counts when dealing with implicitly represented parses. Furthermore, because the permutations correspond to the same parse, they must both belong to the same symmetry class where at least one is encoded implicitly.

Therefore, all parses are accounted for and none are over-represented by either the multiset permutations or the implicit symmetry counts.

4.3.4.1 Complexity

Given that Algorithm 2, which outlines the parsing procedure, just visits each vertex and edge exactly once and neither the vertex- or edge-matching subroutines depends on the yield, complexity is linear in the size of the tree. However, this result is somewhat deceptive given that so much of the work is being done by Algorithm 3, whose complexity is entirely dependent on the grammar. In fact, taking the grammar back into consideration, the naive implementation, which simply assumes *permute*$(E_1, \ldots, E_n)$ is the set of permutations without consideration for duplicate symbols among the E_i, results in a complexity of $O(|\mathcal{R}| \cdot n! \cdot |m|)$, where $|\mathcal{R}|$ is the number of rules in the grammar and n is the maximum number of nonterminals on the right-hand side of any rule.

Considering the rate of growth of the factorial function, the $n!$ term can easily dominate in practice, motivating the optimization described in Section 4.3.1 which exploits symmetries in the rule and limits consideration to the unique permutations of the multiset of nonterminal symbols among the E_i, of which there are $\frac{n!}{\Pi_{j=1}^{k} n_j!}$ where k is the number of distinct symbols among the E_i and n_j is the number of occurrences of a particular symbol among these k. The count of multiset permutations is at its largest when there are equal numbers of each symbol leading to an upper bound of $\frac{n!}{\lceil \frac{n}{k} \rceil !^k}$. Thus, the compact encoding optimization produces a total complexity bound of $O(|\mathcal{R}| \cdot \frac{n!}{\lceil \frac{n}{k} \rceil !^k} \cdot |m|)$.

However, it is important to note that the optimization only holds when there are symmetries in the rule; if it is completely asymmetric (i.e., the nonterminal symbols

associated with edges $E_1, \ldots, E_n$ are all distinct), we are stuck with the original bound of $O(|\mathcal{R}| \cdot n! \cdot |m|)$. On the other hand, the algorithm is relatively fast if the trees have a low branching factor k. In the best case, where $k = 1$, there is only one distinct nonterminal symbol among the $E_1, \ldots, E_n$, and the vertex-matching algorithm need only consider a single permutation for each rule since all the others produce symmetric parses, leading to a bound of $O(|\mathcal{R}| \cdot |m|)$. Thus, grammar design features such as the number of rules, rule size, and how symmetric each rule is, all have important impacts on performance, where rule symmetries in particular can play a crucial role in parsing feasibility.

4.3.5 Relation to graph parsing

Many of the details of unordered tree parsing are also applicable to HRG parsing in general. For instance, just as the unordered tree parser must explore the permutations of nonterminals in the rule right hand side trees, so too must a general HRG parser explore the isomorphisms of right hand side hypergraphs. Consequently, HRG parsers are also plagued by symmetric parses and can benefit from the compact parse encoding scheme described for unordered tree parses, the difference being that instead of ordinary permutations, a HRG parser must count graph isomorphisms, a generalization that reduces to tree permutations in the case of our unordered tree restriction of HRG. In fact, during one of our experiments described in Chapter 8, we tested the general HRG parser described by Chiang et al. (2013) both with and without this compact symmetric parse encoding scheme and found that it reduces memory consumption from a prohibitive 27GB or so to approximately 5GB. Similarly, one can define a canonical form for general HRG rule right-hand-side graphs, allowing for indexing just as in the tree case. We are unaware of any prior paper that details these optimizations for either unordered tree or general graph parsing, however.

Similarly, and potentially more seriously, we are unaware of any HRG parsing paper that discusses how to correctly account for duplicate parses (Section 4.3.2), and because the duplicate parses are a side effect of the lack of ordering on the yield, something unfamiliar from string parsing, even experts in string parsing are unlikely to immediately notice the problem. Most existing graph parsing papers are primarily theoretical and aim mainly to demonstrate polynomial time membership checking algorithms rather than to construct parses per se, leaving the details of how one builds a complete parse forest to the reader. Even Chiang et al. (2013), whose interests were, in fact, in building explicit parse forests, did not address this issue even though it has

potentially profound impact on any probabilistic model that relies on the parser for inference.

Beyond these HRG general implications, the parser also enjoys increased efficiency due to the restriction to tree-shaped graphs. For instance, the time and space complexity of the graph parsing algorithm of Chiang et al. (2013) is, in general, $O((3^d \cdot |m|)^{k+1})$ and $O((2^d \cdot |m|)^{k+1})$, where d is the maximum degree of any vertex in the graph and k is the graph's *treewidth*, another quantity closely related to the density of the graph. For our purposes, d is a constant of the grammar, and the treewidth of a tree-shaped graph is just 1, so the algorithm has both a time and space complexity of $O(|m|^2)$ as compared to our tree-specific parser's linear bound.

Furthermore, this is only the asymptotic analysis. Much of the implementation work of Chiang et al. (2013) algorithm deals with tracking the boundary of the subgraph dominated by each node of the parse tree, analgous to string spans in CFG parsing. Just as in CFG parsing, where combining two parse items under a new parent node in the parse is only valid if the two spans (subgraphs) are disjoint, and if they are contiguous. However, it is more difficult to guarantee these two conditions in general graphs, a detail that accounts for much of the implementational challenge of Chiang et al. (2013). However, in the case of our grammars, a subtree can be identified by a single vertex, its root, saving us a great deal of memory overhead, and we need not conduct any explicit intersection or contiguity testing because we join all subtrees under a given node simultaneously.

4.4 Tree-string synchronous parsing

If utterance meanings are represented by trees, mapping between meaning and utterances can be modeled as a synchronous tree-string grammar that ties together a monolingual unordered tree grammar for parsing the meaning and a monolingual context free string grammar for modeling the utterance. Parsing in this scenario can be formulated as a two stage parsing problem where first we parse the tree using the tree portion of the grammar rules and then subsequently prune the resultant parse forest to only include parses that are also consistent with the string portion of the input. Each parse item of the tree parsing algorithm corresponds to a rule of the original grammar where the tree portion yields some subtree of the input by means of one or more partial derivations in set $\mathcal{X}$. To convert such a monolingual parse item to a parse item of a tree-string synchronous parse, the parser must find at least one of these partial deriva-

tions in $\mathcal{X}$ that also yields some substring consistent with the input string. This process can be implemented in a straightforward manner by converting the parse items of the tree parsing stage into a projection grammar as described in Section 3.3.2, thereby encoding $\mathcal{X}$ as a grammar, and then parsing the string using this new grammar.

Crucial to the complexity analysis, the tree parser produces a packed parse forest requiring memory linear in the tree. The vertex- and edge-matching algorithms output the set of all possible matches for each vertex and edge, and treating the grammar as constant, there are $O(1)$ per vertex/edge (leading to the linear bound for the entire tree). Converting the parse forest to a projection grammar, each match corresponds to a single rule. Thus, assuming an input tree of size $|m|$, the first stage yields a projection grammar with $O(|m|)$ rules, in time which is also linear in $|m|$. The second stage then takes this projection grammar and uses the string portion of the rules to parse the utterance which is of length $|w|$. Any string parsing algorithm capable of handling rules of arbitrary size such as Earley's algorithm (Earley, 1970) can be adapted for the purpose, which in general has time $O(|\mathcal{R}| \cdot |w|^3)$, where $|\mathcal{R}|$ is the number of rules in the string grammar. Using the projection grammar of the first stage, then, the second stage of the synchronous parsing algorithm has complexity $O(|m| \cdot |w|^3)$. Overall time complexity includes the cost of both stages, where tree parsing only takes time $O(|m|)$ and disappears from the asymptotic analysis as it is dominated by the second, string-parsing stage.

In generating the projection grammar, to accommodate asynchronous monolingual rules of the type described in Section 3.3.1, the parser must do some additional work just prior to the string parsing stage. This work basically entails adding any string-only monolingual rules from the original grammar whose left-hand-side nonterminal appears on the right-hand side of any rule in the projection grammar (repeated until there are no more monolingual rules to add). The number of these rules is dependent only on the original grammar and adding them to the projection grammar has no effect on the asymptotic complexity analysis. Downstream, in the string-parsing stage, these extra rules produce an additional $O(|w|^3)$ time complexity, which, like the first stage parsing cost, also disappears in the final asymptotic analysis.

Some complications arise, however, when applying the compact parse encoding scheme described in Section 4.3.1 since monolingual symmetries may not hold when one considers the rule as a whole, requiring some additional handling.

4.4.1 Symmetries in synchronous grammars

Because synchronous nonterminals shared across multiple elements of a rule right hand side can break symmetries, we need to be careful about how we handle parse symmetries, particularly if the synchronous grammar describes a language of ordered structures such as strings. That is, a tree-string synchronous grammar will need to be treated with additional care, since nonterminals that appear in both the tree and string portions of a rule right hand side can no longer be reordered freely.

In translating the tree in Figure 4.1, for instance, we might apply a synchronous grammar rule such as the following.

$$\text{EVENT} \rightarrow \langle\ \lambda e.\text{look}(e) \wedge \text{ROLE}_{\boxed{1}}(e) \wedge \text{ROLE}_{\boxed{2}}(e)\ \big\|\ \text{ROLE}_{\boxed{1}}\ \text{looked}\ \text{ROLE}_{\boxed{2}}\ \rangle$$

When considering the unordered tree alone, there are potentially two symmetric matches. So if we were only interested in monolingual parsing, we could omit one of the parses and simply multiply the probabilities of the other by two when computing total probability mass. However, the string portion of the rule breaks the symmetry due to the ordering on $\text{ROLE}_{\boxed{1}}$ and $\text{ROLE}_{\boxed{2}}$. Ignoring the string portion of the rule and treating it as a monolingual unordered tree grammar rule, it does not matter which of $\text{ROLE}_{\boxed{1}}$ or $\text{ROLE}_{\boxed{2}}$ maps to

$$\text{agent}(e,x) \wedge \text{dog}(x)$$

and which to

$$\text{theme}(e,y) \wedge \text{frog}(y) \wedge \text{experiencer}^{-1}(y,s) \wedge \text{happy}(s) \wedge \text{loc}(y,z) \wedge \text{jar}(z).$$

In translating these expressions into words, one might expect the first to translate as something like “the dog” and the second as “at the happy frog in the jar” to arrive at a the sentence “the dog looked at the happy frog in the jar” – assuming $\text{ROLE}_{\boxed{1}}$ generates “the dog” and $\text{ROLE}_{\boxed{2}}$ generates “the happy frog...” In contrast to the unordered tree case, however, the result on the string side is very different if $\text{ROLE}_{\boxed{1}}$ were to generate “the happy frog...” and ROLE2 “the dog”. That is, while in the monolingual unordered tree case, each nonterminal only generates a subtree of the unordered tree which can be generated in any order without changing the yield, but in the synchronous case each nonterminal generates a subtree-substring pair. While subtrees can be reordered, substrings cannot, thereby constraining the possible mappings and eliminating the symmetry. Thus, when computing the number of symmetries during

synchronous parsing we need to consider all components of the right hand side of the rule.

In synchronous grammars that link unordered with ordered formalisms (such as HRG and CFG), one may lose much of the benefit of the compact symmetric parse encoding scheme, but there is still a potential benefit if the grammar contains monolingual rules like those described in Section 3.3.1. Consider a rule where there are monolingual "background" nonterminals BG.

$$\begin{aligned}\text{EVENT} \rightarrow \langle\ \lambda e.\text{look}(e) \wedge \text{ROLE}_{\boxed{1}}(e) \wedge \text{ROLE}_{\boxed{2}}(e) \wedge \text{BG}(e) \wedge \text{BG}(e) \\ \Big\|\ \text{ROLE}_{\boxed{1}}\ \text{looked}\ \text{ROLE}_{\boxed{2}}\ \rangle\end{aligned}$$

A rule like this might be useful if the fully specified event has four roles, perhaps a location and temporal specifier in addition to the agent and theme, but where only the agent and theme are expressed in the sentence. In this case, just as before, any symmetries involving the two ROLE in the tree are broken by the ordering in the string, but the background nonterminals BG only appear in the unordered tree and thus there may be symmetric parses even during synchronous parsing.

We can encode the permutations of the monolingual nonterminals just as in the fully monolingual setting even as we explicitly enumerate the permutations for the shared, conventional synchronous nonterminals such as the ROLE nonterminals in the example rule. We still multiply rules by the symmetry count $\frac{\prod_{i=1}^{k} n_i!}{\prod_{i=1}^{\ell} \tau_i!}$ but where the n_i and τ_i counts are limited to the monolingual nonterminals.

4.5 Conclusion

The unordered tree grammars defined at the beginning of this chapter combine features of both hyperedge replacement grammar and regular tree grammars. The partial ordering of general directed graphs lends the formalism the ability to generate and parse languages of unordered trees, a feature that is useful for working with tree representations of a subclass of predicate calculus expressions. At the same time, the similarities to regular tree grammar permit a far more efficient parsing algorithm, one that is linear in the size of the tree, an efficiency which will prove critical for exploring a highly ambiguous space of candidate meanings such as those licensed by the scene graphs described in Chapter 7. Furthermore, because the grammars have context-free derivations they are compatible with context-free string grammars, facilitating integration into a synchronous grammar for jointly modeling meaning and sentence pairs for semantic

parsing (or scene-meaning-sentence triples for word learning). Specifically, training a semantic parser based on a synchronous unordered tree-string grammar would have a time complexity of $O(|m| \cdot |w|^3)$ where $|m|$ is the size of the meaning representation and $|w|$ the size of the sentence, or $O(|w|^4)$ if one assumes $|m| \approx |w|$. Parsing is therefore more expensive than typical for conventional monolingual syntactic parsing but comparable to other approaches to semantic parsing such as Wong (2007) and less than others such as the $O(|w|^7)$ time complexity of Kwiatkowski et al. (2010). Additionally, the grammar class and algorithmic innovations outlined are more generally applicable than those employed by many other semantic parsers. In fact, many of the optimizations and implementation details apply equally well to HRG parsing in general, increasing the generality of the innovations described in this chapter to a much wider class of languages which may be valuable for many other applications.

Chapter 5

Inference in Multi-weighted Grammars

Any grammar formalism with context free derivation trees can be used to define a probabilistic model by assigning weights. For instance, PCFGs do this by assigning weights to each rule such that the sum of the weights of all rules with a given left hand side is one. In particular, the model thus defined is a product of multinomials, where the probability of a derivation tree is a product of the weights of its constituent rules. That is, given a derivation tree x, its yield y and rule weights θ, we have the following probability:

$$p(y,x|\theta) = \prod_{r\in\mathcal{R}} \theta(r)^{n_r(x)}$$

or if there are multiple items to be parsed (say, N), as is more commonly the case,

$$p(\mathbf{y},\mathbf{x}|\theta) = \prod_{i=1}^{N}\prod_{r\in\mathcal{R}} \theta(r)^{n_r(x_i)}, \tag{5.1}$$

where r is a rule of the grammar, $\theta(r)$ is its weight, and $n_r(x)$ is the number of times r appears in x.

Recall our definition for multi-weighted grammars in Section 3.4, where each rule weight $\theta(r)$ is factored into a sequence of weights $\omega(r) = \omega_1(r)\omega_2(r)...\omega_n(r)$, and each scalar weight $\omega_i(r)$ is a single parameter of a multinomial identified by the conditioning information c in feature pair $\varphi_i(r) = \langle c,e\rangle$. Then Equation 5.1 holds equally well as for PCFGs, and inference is consequently very similar, but we must pay some care to the factors $\omega_i(r)$. Before we proceed, it is useful to list a few definitions, which will make the following derivation flow more smoothly. First, we define $\mathcal{F}$ as the set of all feature pairs associated with any rule, $\mathcal{F}_c$, a subset of $\mathcal{F}$ where all feature pairs share the same conditioning information c, and $\mathcal{C}$, the set of features used as conditioning

information.

$$\mathcal{F} := \{\langle c, e\rangle = \varphi_i(r) : i \in \mathbb{Z}, r \in \mathcal{R}\}$$
$$\mathcal{F}_c := \{\langle c, e\rangle \in \mathcal{F}\}$$
$$\mathcal{C} := \{c : \langle c, e\rangle \in \mathcal{F}\}$$

Every factor in a multi-weighted grammar is defined as a conditional probability, $P(e|c)$, where $P(e|c)$ is a multinomial with a set of weights as parameters. This set and the individual weights are denoted by θ_c and $\theta_c(e)$, respectively.

$$\theta_c := \{\omega_i(r) : \varphi_i(r) \in \mathcal{F}_c\},$$
$$\theta_c(e) := \omega_i(r) \in \theta_c \text{ where } \varphi_i(r) = \langle c, e\rangle \text{ for any } r \in \mathcal{R}, i \in \mathbb{Z}$$

That is, $\theta_c(e) = P(e|c)$ is the specific weight for pair $\langle c, e\rangle$, and θ_c is the full set of all parameters for $P(\cdot|c)$. It will also sometimes be useful to refer to the full set of all multinomials:

$$\theta := \{\theta_c : c \in \mathcal{C}\}.$$

Using this notation, Equation 5.1 can be expressed in terms of feature pairs:

$$\begin{aligned} p(\mathbf{y}, \mathbf{x}|\theta) &= \prod_{i=1}^{N} \prod_{r \in \mathcal{R}} \theta(r)^{n_r(x_i)} \\ &= \prod_{i=1}^{N} \prod_{r \in \mathcal{R}} \prod_{\langle c,e\rangle \in \varphi(r)} \theta_c(e)^{n_r(x_i) n_{c,e}(\varphi(r))} \\ &= \prod_{i=1}^{N} \prod_{\langle c,e\rangle \in \mathcal{F}} \theta_c(e)^{n_{c,e}(x_i)} \end{aligned} \tag{5.2}$$

where $n_{c,e}(r)$ and $n_{c,e}(x)$ are, respectively, the number of instances of feature pair $\langle c, e\rangle$ in rule r and in all the rules of derivation tree x. Note that this feature pair-based form reduces to Equation 5.1 if each rule only has a single feature pair consisting of the rule itself paired with the left-hand side symbol as the conditioning information.

In the expectationmaximization (EM) algorithm, we are typically interested in finding the point estimate for θ that maximizes $p(\mathbf{y}|\theta)$, where $\mathbf{y}$ is the vector of N items in the training data (strings in syntactic parsing or graph-tree pairs in a synchronous grammar-based semantic parser, for example). In the Bayesian setting, however, we place a prior over each of the multinomial parameter sets θ_c and estimate a posterior

distribution over both the vector of derivations $\mathbf{x}$ and the full set of multinomial parameters θ.

$$p(\theta, \mathbf{x}|\mathbf{y}, \alpha) = \frac{p(\mathbf{y}, \mathbf{x}, \theta|\alpha)}{p(\mathbf{y}|\alpha)} = \frac{\prod_{c \in \mathcal{C}} p(\theta_c|\alpha_c) \prod_{i=1}^{N} p(y_i, x_i|\theta)}{\int \prod_{c \in \mathcal{C}} p(\theta_c|\alpha_c) \prod_{i=1}^{N} \sum_{x \in \mathcal{X}_i} p(y_i, x|\theta) d\theta} \tag{5.3}$$

One particularly popular choice for the prior $p(\theta_c|\alpha_c)$ is the Dirichlet distribution:

$$\mathcal{D}(\theta_c|\alpha_c) = \frac{\Gamma(\sum_{\langle c,e \rangle \in \mathcal{F}_c} \alpha_c(e))}{\prod_{\langle c,e \rangle \in \mathcal{F}_c} \Gamma(\alpha_c(e))} \prod_{\langle c,e \rangle \in \mathcal{F}_c} \theta_c(e)^{\alpha_c(e)-1}. \tag{5.4}$$

Like θ, the Dirichlet parameters α is a set of parameter settings, one parameter set α_c per θ_c. While $\theta_c(e)$ is the particular weight in θ_c associated with feature e, each α_c is a set of positive real valued pseudocounts for the number of occurrences of features, including that of e denoted by $\alpha_c(e)$. If each $\alpha_c(e) \geq 1$, the Dirichlet can be interpreted as defining the probability over a particular assignment to the weights $\theta_c(e)$ given that each has been seen $\alpha_c(e) - 1$ times. When the pseudocounts are $0 < \alpha_c(e) < 1$, the Dirichlet defines a sparse prior that assigns high probability to weight vectors with only a few weights having large values. This can be useful for approximating power law distributions, which are endemic to natural language, and can also be helpful for unsupervised learning where there is often a large set of possible rules but only a few are useful and we do not know a priori which ones they may be.

The Dirichlet is also notable as the *conjugate prior* to the multinomial. This means that given a Dirichlet prior over multinomial parameters θ_c, the posterior probability of θ_c given observations x is also Dirichlet. That is,

$$p(x, \theta_c|\alpha_c) = p(x|\theta_c)\mathcal{D}(\theta_c|\alpha_c) \implies p(\theta_c|x, \alpha_c) = \mathcal{D}(\theta_c|\hat{\alpha}_c) \tag{5.5}$$

where $\hat{\alpha}_c(e) = n_{c,e}(x) + \alpha_c(e)$. As we will see, this property greatly simplifies inference.

In spite of this nice property of the Dirichlet, however, a problem often arises in Bayesian inference, and our situation is no different, where the full posterior (Equation 5.3) proves to be intractable. This has lead to a number of strategies that employ approximations that are more readily computed, such as sampling or variational Bayes (VB). We employ the latter strategy, making use of the popular "mean field" assumption to limit the space of solutions to a tractable set. The basic approach of VB has been outlined by Bishop (2006), who presents a high level introduction to variational

Bayes. Our algorithm is very similar to that of Kurihara and Sato (2006), who present a mean field-based VB algorithm for PCFGs. In fact, the approach closely resembles that commonly taken with models based on products of multinomials such as HMMs, a special case of PCFGs, which was worked out in detail by Beal (2003), or the popular latent Dirichlet allocation model (Blei, 2004).

In this chapter we derive a batch learning algorithm for optimizing a lower bound on the joint posterior probability of the parses and data and Section 5.4 describes the algorithm itself that closely resembles the Expectation Maximization algorithm for unsupervised grammar rule weight estimation. Since our ultimate purpose is to simulate a human learner, some may object that a batch learner, which learns by repeatedly iterating over the entire data set, is less appropriate than an iterative learner which visits each training item exactly once, updating parameters at each step. However, our primary interest is in the model itself rather than any effects the training algorithm or data set size may have on learning performance. A batch algorithm usually does a better job of exploiting a small data set to optimize the learning objective, making it a better choice for testing the model since a less optimal learning procedure could confound model properties with those of the learning procedure. Furthermore, it is often straightforward to adapt a batch learning algorithm for an incremental learning procedure. Kwiatkowski et al. (2012), for example, describe an online Variational Bayesian Expectation Maximization algorithm, essentially an incremental variant of the algorithm described by Kurihara and Sato (2006). In fact, our own algorithm can be adapted in exactly the same way, so researchers interested in exploring learning progressions, measuring changes training item by training item can easily make the necessary changes.

5.1 The mean field approximation $q(\theta, \mathbf{x}) = q(\theta)q(\mathbf{x})$

Since the integral in the denominator of the expression for the posterior in Equation 5.3 is intractable, we look for an appropriate approximation $q(\theta, \mathbf{x}) \approx p(\theta, \mathbf{x}|\mathbf{y}, \alpha)$. In particular, we assume the feature weights and the derivations are independent, i.e., $q(\theta, \mathbf{x}) = q_\theta(\theta)q_\mathbf{x}(\mathbf{x})$. The basic idea is then to define a lower bound $\hat{\mathcal{L}}[q] \leq \ln p(\mathbf{y}, \mathbf{w}|\alpha)$ in terms of q and then apply the calculus of variations to find a q that

maximizes $\hat{\mathcal{L}}[q]$.

$$\begin{aligned}
\ln p(\mathbf{y}|\alpha) &= \ln \int p(\theta|\alpha)p(\mathbf{y},\mathbf{x}|\theta)d\theta d\mathbf{x} \\
&= \ln \int q(\theta,\mathbf{x})\frac{p(\theta|\alpha)p(\mathbf{y},\mathbf{x}|\theta)}{q(\theta,\mathbf{x})}d\theta d\mathbf{x} \\
&= \ln E_q\left[\frac{p(\theta|\alpha)p(\mathbf{y},\mathbf{x}|\theta)}{q(\theta,\mathbf{x})}\right] \\
&\geq E_q\left[\ln\frac{p(\theta|\alpha)p(\mathbf{y},\mathbf{x}|\theta)}{q(\theta,\mathbf{x})}\right].
\end{aligned}$$

The last step is arrived at through Jensen's Inequality, and this final quantity is our lower bound $\hat{\mathcal{L}}[q]$ which we are to maximize.[1] Equivalently, this maximization is often visualized instead as a minimization of the quantity

$$\ln p(\mathbf{y}|\alpha) - E_q\left[\ln\frac{p(\theta|\alpha)p(\mathbf{y},\mathbf{x}|\theta)}{q(\theta,\mathbf{x})}\right] = -E_q\left[\ln\frac{p(\theta|\alpha)p(\mathbf{x}|\mathbf{y},\theta)}{q(\theta,\mathbf{x})}\right]$$

which is the Kullback-Leibler divergence $KL(q\|p)$. That is, maximizing the lower bound is the same as minimizing the KL divergence between our approximation and the true posterior.

Simplifying the lower bound by applying our independence assumption yields the following formula:

$$\begin{aligned}
\hat{\mathcal{L}}[q] &= E_q[\ln p(\mathbf{y},\mathbf{x}|\theta)] + E_q[\ln p(\theta|\alpha)] - E_q[\ln q(\theta,\mathbf{x})] \\
&= E_q[\ln p(\mathbf{y},\mathbf{x}|\theta)] + E_{q_\theta}[\ln p(\theta|\alpha)] - E_{q_\theta}[\ln q(\theta)] - E_{q_\mathbf{x}}[\ln q(\mathbf{x})]. \qquad (5.6)
\end{aligned}$$

We can optimize functional $\hat{\mathcal{L}}[q]$ subject to the constraints that $q(\theta)$ and $q(\mathbf{x})$ both integrate to one (they should be probability distributions) using the method of Lagrange multipliers, with the Lagrange function:

$$\begin{aligned}
\ell[q] = E_q[\ln p(\mathbf{y},\mathbf{x}|\theta)] - E_{q_\mathbf{x}}[\ln q(\mathbf{x})] + \lambda_\mathbf{x}\left(\int q(\mathbf{x})d\mathbf{x} - 1\right) \\
+ E_{q_\theta}[\ln p(\theta|\alpha)] - E_{q_\theta}[\ln q(\theta)] + \lambda_\theta\left(\int q(\theta)d\theta - 1\right).
\end{aligned}$$

[1] The problem formulation may be familiar form EM. However, in the case of EM, the objective is different because there is no prior over θ, leading to a slightly different lower bound

$$E_{q(\mathbf{x})}\left[\ln\frac{p(\mathbf{y},\mathbf{x}|\theta)}{q(\mathbf{x})}\right] \qquad \text{(EM objective)}$$

where $q(\mathbf{x})$ is just $p(\mathbf{x}|\mathbf{y},\theta)$ with a particular assignment for θ.

Then we have the following derivatives:

$$\frac{\delta \ell}{\delta q_\theta} = E_{q_\mathbf{x}}[\ln p(\mathbf{y},\mathbf{x}|\theta)] + \ln p(\theta|\alpha) - \ln q(\theta) - 1 + \lambda_\theta \qquad \frac{d\ell}{d\lambda_\theta} = \int q(\theta) d\theta - 1$$
$$\frac{\delta \ell}{\delta q_\mathbf{x}} = E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)] - \ln q(\mathbf{x}) - 1 + \lambda_\mathbf{x} \qquad \frac{d\ell}{d\lambda_\mathbf{x}} = \int q(\mathbf{x}) d\mathbf{x} - 1$$

Setting the derivatives to zero and solving for q_θ and $q_\mathbf{x}$ yields:

$$q_\theta(\theta) = \frac{p(\theta|\alpha)\exp\left(E_{q_\mathbf{x}}[\ln p(\mathbf{y},\mathbf{x}|\theta)]\right)}{\int p(\theta|\alpha)\exp\left(E_{q_\mathbf{x}}[\ln p(\mathbf{y},\mathbf{x}|\theta)]\right) d\theta}, \tag{5.7}$$

$$q_\mathbf{x}(\mathbf{x}) = \frac{\exp\left(E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)]\right)}{\int \exp\left(E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)]\right) d\mathbf{x}}. \tag{5.8}$$

5.2 Deriving $q(\theta)$

Plugging in the definitions for $p(\theta|\alpha)$ and $p(\mathbf{y},\mathbf{x}|\theta)$ in Equations 5.4 and 5.2, respectively, results in the following expression for the expectation $E_{q_\mathbf{x}}[\ln p(\mathbf{y},\mathbf{x}|\theta)]$:

$$\begin{aligned}
E_{q_\mathbf{x}}[\ln p(\mathbf{y},\mathbf{x}|\theta)] &= E_{q_\mathbf{x}}\left[\ln \prod_{i=1}^{N} p(y_i,x_i|\theta)\right] \\
&= \sum_{i=1}^{N} E_{q_\mathbf{x}}[\ln p(y_i,x_i|\theta)] \\
&= \sum_{i=1}^{N} E_{q_\mathbf{x}}\left[\ln \prod_{\langle c,e\rangle \in \mathcal{F}} \theta_c(e)^{n_{c,e}(x_i)}\right] \\
&= \sum_{\langle c,e\rangle \in \mathcal{F}} \left(\sum_{i=1}^{N} E_{q_\mathbf{x}}[n_{c,e}(x_i)]\right) \ln \theta_c(e) \\
&= \ln \prod_{\langle c,e\rangle \in \mathcal{F}} \theta_c(e)^{\sum_{i=1}^{N} E_{q_\mathbf{x}}[n_{c,e}(x_i)]} \\
&= \ln \prod_{c \in \mathcal{C}} \prod_{\langle c,e\rangle \in \mathcal{F}_c} \theta_c(e)^{\sum_{i=1}^{N} E_{q_\mathbf{x}}[n_{c,e}(x_i)]}
\end{aligned} \tag{5.9}$$

Note that although we only assumed independence between θ and $\mathbf{x}$, the factorization of $E_{q_\mathbf{x}}[\ln p(\mathbf{y},\mathbf{x}|\theta)]$ produces an even stronger independence result. Specifically, $q_\theta(\theta)$ can be expressed as a product of independent probabilities q_{θ_c} by combining Equations 5.7 and 5.9.

$$q_\theta(\theta) = \prod_{c \in \mathcal{C}} \frac{p(\theta_c|\alpha_c)\prod_{\langle c,e\rangle \in \mathcal{F}_c} \theta_c(e)^{\sum_{i=1}^{N} E_{q_\mathbf{x}}[n_{c,e}(x_i)]}}{\int p(\theta_c|\alpha_c)\prod_{\langle c,e\rangle \in \mathcal{F}_c} \theta_c(e)^{\sum_{i=1}^{N} E_{q_\mathbf{x}}[n_{c,e}(x_i)]} d\theta_c} = \prod_{c \in \mathcal{C}} q_{\theta_c}(\theta_c)$$

Simplifying q_{θ_c} yields

$$q_{\theta_c}(\theta_c) = \frac{p(\theta_c|\alpha_c)\prod_{\langle c,e\rangle\in\mathcal{F}_c}\theta_c(e)^{\sum_{i=1}^{N}E_{q_\mathbf{x}}[n_{c,e}(x_i)]}}{\int p(\theta_c|\alpha_c)\prod_{\langle c,e\rangle\in\mathcal{F}_c}\theta_c(e)^{\sum_{i=1}^{N}E_{q_\mathbf{x}}[n_{c,e}(x_i)]}d\theta_c}$$

By introducing a change of variables

$$\hat{\alpha}_c(e) = \alpha_c(e) + \sum_{i=1}^{N} E_{q_\mathbf{x}}\left[n_{c,e}(x_i)\right] \tag{5.10}$$

and making use of the definition of the Dirichlet distribution in Equation 5.4 we arrive at the final simplification:

$$\begin{aligned} q_{\theta_c}(\theta_c) &= \frac{B(\alpha)^{-1}\prod_{\langle c,e\rangle\in\mathcal{F}_c}\theta_c(e)^{\hat{\alpha}_c(e)-1}}{B(\alpha)^{-1}\int\prod_{\langle c,e\rangle\in\mathcal{F}_c}\theta_c(e)^{\hat{\alpha}_c(e)-1}d\theta} \\ &= B(\hat{\alpha})^{-1}\prod_{\langle c,e\rangle\in\mathcal{F}_c}\theta_c(e)^{\hat{\alpha}_c(e)-1}. \end{aligned} \tag{5.11}$$

Here, the beta function $B(\alpha) = \int\prod_{\langle c,e\rangle\in\mathcal{F}_c}\theta_c(e)^{\alpha_c(e)-1}d\theta$ is the partition function for the Dirichlet distribution.

Thus, $q_{\theta_c}(\theta_c)$ is also Dirichlet with parameters $\hat{\alpha}_c$. We have essentially just re-derived the proof of Dirichlet-Multinomial conjugacy sketched in Equation 5.5.

5.3 Deriving $q(\mathbf{x})$

All that is left is to find the optimal variational distribution over derivation trees $q(\mathbf{x})$, which we note from Equation 5.8 is defined in terms of the expectation $E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)]$. The definition of $p(\mathbf{y},\mathbf{x}|\theta)$ in Equation 5.2 takes us most of the way.

$$\begin{aligned} E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)] &= E_{q_\theta}\left[\ln\prod_{i=1}^{N}\prod_{\langle c,e\rangle\in\mathcal{F}}\theta_c(e)^{n_{c,e}(x_i)}\right] \\ &= E_{q_\theta}\left[\ln\prod_{i=1}^{N}\prod_{c\in\mathcal{C}}\prod_{\langle c,e\rangle\in\mathcal{F}_c}\theta_c(e)^{n_{c,e}(x_i)}\right] \\ &= \sum_{i=1}^{N}\sum_{c\in\mathcal{C}}\sum_{\langle c,e\rangle\in\mathcal{F}_c} n_{c,e}(x_i)E_{q_{\theta_c}}[\ln\theta_c(e)]. \end{aligned}$$

Making the variable substitution $\hat{\theta}_c(e) = \exp\left(E_{q_{\theta_c}}[\ln\theta_c(e)]\right)$ we get

$$\begin{aligned} E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)] &= \sum_{i=1}^{N}\sum_{c\in\mathcal{C}}\sum_{\langle c,e\rangle\in\mathcal{F}_c} n_{c,e}(x_i)\ln\hat{\theta}_c(e) \\ &= \ln\prod_{i=1}^{N}\prod_{c\in\mathcal{C}}\prod_{\langle c,e\rangle\in\mathcal{F}_c}\hat{\theta}_c(e)^{n_{c,e}(x_i)}. \end{aligned} \tag{5.12}$$

Finally, plugging this expression for $E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)]$ into the formula previously derived for $q(\mathbf{x})$ (Equation 5.8) produces

$$q_{\mathbf{x}}(\mathbf{x}) = \prod_{i=1}^{N} \frac{\prod_{c\in\mathcal{C}}\prod_{\langle c,e\rangle\in\mathcal{F}_c}\hat{\theta}_c(e)^{n_{c,e}(x_i)}}{\sum_{x\in\mathcal{X}_i}\prod_{c\in\mathcal{C}}\prod_{\langle c,e\rangle\in\mathcal{F}_c}\hat{\theta}_c(e)^{n_{c,e}(x_i)}} = \prod_{i=1}^{N} q_{x_i}(x_i) \tag{5.13}$$

which, like $q_\theta(\theta)$, can be expressed in terms of a product of separate probability distributions over the derivation trees of the individual y_i.

To compute $\hat{\theta}$ we make use of a standard result for the Dirichlet distribution[2]:

$$E_{q_{\theta_c}}[\ln\theta_c(e)] = \Psi(\hat{\alpha}_c(e)) - \Psi\Big(\sum_{\langle c,e\rangle\in\mathcal{F}_c}\hat{\alpha}_c(e)\Big) \implies$$

$$\hat{\theta}_c(e) = \exp\left(E_{q_{\theta_c}}[\ln\theta_c(e)]\right) = \exp\left(\Psi(\hat{\alpha}_c(e)) - \Psi\Big(\sum_{\langle c,e\rangle\in\mathcal{F}_c}\hat{\alpha}_c(e)\Big)\right). \tag{5.14}$$

These $\hat{\theta}$ parameters are sub-normalized, i.e., $\sum_{\langle c,e\rangle\in\mathcal{F}_c}\hat{\theta}_c(e) \leq \sum_{\langle c,e\rangle\in\mathcal{F}_c}\theta_c(e) = 1$, shown by Jensen's Inequality:

$$\hat{\theta}_c(e) = \exp\left(E_{p(\theta_c|\alpha_c)}[\ln\theta_c(e)]\right) \leq \exp\left(\ln E_{p(\theta_c|\alpha_c)}[\theta_c(e)]\right) = E_{p(\theta_c|\alpha_c)}[\theta_c(e)].$$

The distribution $q(x_i)$ is the variational counterpart to $p(x_i|y_i,\theta)$, which is the posterior probability of a derivation tree given the rule weights θ and a particular data item y_i. In fact, $q(x_i)$ takes exactly the same form, except that $\hat{\theta}$ is substituted for the multinomial parameters of $p(x_i,y_i|\theta)$. Because $\hat{\theta}$ is sub-normalized, $q(x_i)$ is not a true posterior, but the normalization constant in the denominator guarantees that it is, nonetheless, a genuine probability distribution in its own right. Consequently, it is not necessary or desirable to re-normalize each $\hat{\theta}_c$. In fact, doing so will destroy a particularly felicitous property of the $\Psi(\cdot)$ function, which is clear from the following approximation:

$$\exp(\Psi(\hat{\alpha}_c(r))) \approx \begin{cases} \frac{\hat{\alpha}_c(e)^2}{2} & \text{if } \hat{\alpha}_c(e) \in [0,1] \\ \hat{\alpha}_c(e) - \frac{1}{2} & \text{if } \hat{\alpha}_c(e) > 1 \end{cases}.$$

That is, $\exp(\Psi(\hat{\alpha}_c(e)))$ effectively subtracts $\frac{1}{2}$ from the expected counts while taking care to keep everything well defined by avoiding negative counts when $\hat{\alpha}_c(e)$ is less than $\frac{1}{2}$. This is precisely what allows VB to model sparsity, subtracting counts from rules so that those for which we see very little use get only a very small weight.

[2] See Appendix A for a derivation.

5.4 The variational Bayes inference algorithm

To summarize, we have found the $q(\theta_t)$ and $q(x_i)$ that maximize $\hat{\mathcal{L}}[q]$ by taking derivatives of the Lagrangian, setting them to zero, and solving, yielding the variational distributions

$$q(\theta_c) = \mathcal{D}(\theta_c|\hat{\alpha}_c) \tag{5.11}$$

$$q(x_i) = \frac{\prod_{\langle c,e\rangle\in\mathcal{F}} \hat{\theta}_c(e)^{n_{c,e}(x_i)}}{\sum_{x\in\mathcal{X}_i}\prod_{\langle c,e\rangle\in\mathcal{F}} \hat{\theta}_c(e)^{n_{c,e}(x)}}. \tag{5.13}$$

which have parameters

$$\hat{\alpha}_c(e) = \alpha_c(e) + \sum_{i=1}^{N} E_{q_{x_i}}[n_{c,e}(x_i)] \tag{5.10}$$

$$\hat{\theta}_c(e) = \exp\left(\Psi(\hat{\alpha}_c(e)) - \Psi(\sum_{\langle c,e\rangle\in\mathcal{F}_c} \hat{\alpha}_c(e))\right). \tag{5.14}$$

The parameters of $q(\theta_c)$ are defined in terms of $q(x_i)$ and the parameters of $q(x_i)$ with respect to the parameters of $q(\theta_c)$. When computing the $\hat{\alpha}$ parameters, the inside-outside algorithm efficiently calculates the variational probability of a derivation tree $q(x_i)$. Thus, we can perform an EM-like alternation between calculating $\hat{\alpha}$ and $\hat{\theta}$. [3] Just as we would in EM, we use the inside-outside algorithm to compute the expected counts of the rules, from which we can, in turn, estimate the number of occurrences of each pair $\langle c,e\rangle$, using the relation $n_{c,e}(x) = n_{c,e}(r)\cdot n_r(x)$, where $n_{c,e}(r)$ is the number of times $\langle c,e\rangle$ appears in rule r. This expected count is denoted by by $\hat{\alpha}_c(e)$ in Equation 5.10. Then we use these parameters to estimate the expected values of the $\hat{\theta}$ parameters.

Repeatedly alternating between computing these two expectations, the algorithm eventually converges to a local maximum of the variational lower bound. This is essentially the same as for the conventional EM algorithm for estimating the rule weights of ordinary PCFGs, but in the case of the multi-weighted grammars the $\hat{\theta}$ parameters are identified with individual factors of the rule probabilities $\omega(r)$, rather than complete rule probabilities themselves. In fact, if $\omega(r)$ is a scalar and $f(r)$ just identifies a single feature pair comprised of the rule and its left-hand side, then the algorithm is identical to the PCFG case, and this assumption would result in exactly the same expression

[3] Because of the resemblance to EM, this procedure has been called VBEM. Unlike EM, however, the procedure alternates between computing the expected values of two different sets of variational parameters and lacks a maximization step.

for the objective function presented by Kurihara and Sato (2006) for PCFGs. We will now describe this objective function for the more general case of full multi-weighted grammars.

5.5 The lower bound

For judging whether the procedure has converged, as well as for a number of other things such as model selection, it is useful to have an explicit formula for computing the variational lower bound $\hat{\mathcal{L}}[q]$, sometimes referred to as the evidence lower bound (ELBO), or negative free energy. The general form of the equation for the mean field assumption is Equation 5.6, but it is possible to get a simplification using the formula for $q(\mathbf{x})$ (Equation 5.8).

$$\begin{aligned} E_{q_\mathbf{x}}[\ln q(\mathbf{x})] &= E_{q_\mathbf{x}}\left[\ln \frac{\exp\left(E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)]\right)}{\int \exp\left(E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)]\right) d\mathbf{x}}\right] \\ &= E_q[\ln p(\mathbf{y},\mathbf{x}|\theta)] - \ln \int \exp\left(E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)]\right) d\mathbf{x} \end{aligned}$$

Substituting this for $E_{q_\mathbf{x}}[\ln q(\mathbf{x})]$ in Equation 5.6 yields the following formula for $\hat{\mathcal{L}}[q]$:

$$\begin{aligned} \hat{\mathcal{L}}[q] &= E_q[\ln p(\mathbf{y},\mathbf{x}|\theta)] + E_{q_\theta}[\ln p(\theta|\alpha)] - E_{q_\theta}[\ln q(\theta)] - E_{q_\mathbf{x}}[\ln q(\mathbf{x})] \\ &= \ln \int \exp\left(E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)]\right) d\mathbf{x} + E_{q_\theta}[\ln p(\theta|\alpha)] - E_{q_\theta}[\ln q(\theta)]\,. \end{aligned}$$

Breaking things down term by term results in the following:

$$\begin{aligned} \ln \int \exp\left(E_{q_\theta}[\ln p(\mathbf{y},\mathbf{x}|\theta)]\right) d\mathbf{x} &= \ln \int \prod_{i=1}^{N} \prod_{c\in\mathcal{C}} \prod_{\langle c,e\rangle\in\mathcal{F}_c} \hat{\theta}_c(e)^{n_{c,e}(x_i)} d\mathbf{x} \quad \text{(from Eq. 5.12)} \\ &= \sum_{i=1}^{N} \ln \sum_{x_i\in\mathcal{X}_i} \prod_{c\in\mathcal{C}} \prod_{\langle c,e\rangle\in\mathcal{F}_c} \hat{\theta}_c(e)^{n_{c,e}(x_i)} \\ E_{q_\theta}[\ln p(\theta|\alpha)] &= \sum_{c\in\mathcal{C}} E_{q_{\theta_c}}[\ln p(\theta|\alpha)] \quad \text{(apply Eq. 5.4)} \\ &= \sum_{c\in\mathcal{C}} \ln\Gamma\Big(\sum_{\langle c,e\rangle\in\mathcal{F}_c} \alpha_c(e)\Big) - \sum_{\langle c,e\rangle\in\mathcal{F}_c} \ln\Gamma(\alpha_c(e)) \\ &\quad + \sum_{\langle c,e\rangle\in\mathcal{F}_c} (\alpha_c(e)-1) E_{q_{\theta_c}}[\ln\theta_c(e)] \\ &= \sum_{c\in\mathcal{C}} \ln\Gamma\Big(\sum_{\langle c,e\rangle\in\mathcal{F}_c} \alpha_c(e)\Big) - \sum_{\langle c,e\rangle\in\mathcal{F}_c} \ln\Gamma(\alpha_c(e)) \\ &\quad + \sum_{\langle c,e\rangle\in\mathcal{F}_c} (\alpha_c(e)-1) \ln\hat{\theta}_c(e) \end{aligned}$$

$$E_{q_\theta}[\ln q(\theta)] = \sum_{c\in\mathcal{C}} \ln\Gamma(\sum_{\langle c,e\rangle\in\mathcal{F}_c} \hat{\alpha}_c(e)) - \sum_{\langle c,e\rangle\in\mathcal{F}_c} \ln\Gamma(\hat{\alpha}_c(e)) + \sum_{\langle c,e\rangle\in\mathcal{F}_c} (\hat{\alpha}_c(e)-1)\ln\hat{\theta}_c(e). \quad \text{(from Eq. 5.11)}$$

Finally, putting it all back together again, we arrive at

$$\begin{aligned}\hat{\mathcal{L}}[q] = &\sum_{i=1}^{N} \ln \sum_{x\in\mathcal{X}_i} \prod_{c\in\mathcal{C}} \prod_{\langle c,e\rangle\in\mathcal{F}_c} \hat{\theta}_c(e)^{n_{c,e}(x_i)} \qquad (5.15)\\ &+ \sum_{c\in\mathcal{C}} \ln\Gamma(\sum_{\langle c,e\rangle\in\mathcal{F}_c} \alpha_c(e)) - \sum_{\langle c,e\rangle\in\mathcal{F}_c} \ln\Gamma(\alpha_c(e)) + \sum_{\langle c,e\rangle\in\mathcal{F}_c} (\alpha_c(e)-1)\ln\hat{\theta}_c(e)\\ &\quad - \ln\Gamma(\sum_{\langle c,e\rangle\in\mathcal{F}_c} \hat{\alpha}_c(e)) + \sum_{\langle c,e\rangle\in\mathcal{F}_c} \ln\Gamma(\hat{\alpha}_c(e)) - \sum_{\langle c,e\rangle\in\mathcal{F}_c} (\hat{\alpha}_c(e)-1)\ln\hat{\theta}_c(e).\end{aligned}$$

The first term can be computed by adding the inside "probability" of the start symbol for each training item and is produced as a by-product of the inference algorithm.[4] In fact, it is exactly the same as the quantity computed in the EM setting where there is no prior, but where $\hat{\theta}$ has been substituted for θ.

5.6 Estimating Dirichlet parameters with variational EM

The basic VB algorithm treats the Dirichlet parameters α as fixed quantities, which are assigned manually at the start. However, if we want to automatically estimate the Dirichlet prior parameters, a la Empirical Bayes, we can derive a kind of Variational EM algorithm where the E-step performs the updates for the variational parameters $\hat{\alpha}$ and $\hat{\theta}$, and the M-step maximizes $\hat{\mathcal{L}}[q]$ with respect to α. Although working with a different Dirichlet-Multinomial model, Airoldi et al. (2008) did precisely this, as did Braun and McAuliffe (2010) working with yet another combination of model and prior.

One approach to performing the update in the M-step is to derive a Newton-Raphson algorithm, requiring the first and second derivatives of $\hat{\mathcal{L}}[q]$ (Equation 5.15). For simplicity, assume the Dirichlet priors are symmetric, i.e., $\alpha_c(e) = a_c$ for all $\langle c,e\rangle \in \mathcal{F}_c$, and that $|\mathcal{F}_c| = K_c$. The only terms that depend directly on α are the ones in

$$E_{q_\theta}[\ln p(\theta|\alpha)]] = \sum_{c\in\mathcal{C}} \ln\Gamma(K_c a_c) - K_c \ln\Gamma(a_c) + K_c(a_c-1)\ln\bar{\hat{\theta}}_c(e)$$

where $\ln\bar{\hat{\theta}}_c(e) = \frac{1}{K_c}\sum_{\langle c,e\rangle\in\mathcal{F}_c} \ln\hat{\theta}_c(e)$. This expectation is essentially just that of the log of the likelihood of a set of multinomial parameters under a Dirichlet. Consequently,

[4] Technically, $\sum_{x\in\mathcal{X}_i}\prod_{c\in\mathcal{C}}\prod_{\langle c,e\rangle\in\mathcal{F}_c}\hat{\theta}_c(e)^{n_{c,e}(x_i)}$ is not a proper probability because $\hat{\theta}_c$ is sub-normalized, but the inside-outside algorithm works just the same.

maximizing $\hat{\mathcal{L}}[q]$ reduces to that of estimating the Dirichlet parameters that maximizes this likelihood, which is exactly the problem addressed by Minka (2000) and our solution follows the same form.

Computing the gradient and second derivative,

$$g(a_c) = \frac{d\hat{\mathcal{L}}[q]}{da_c} = K_c \cdot \left(\Psi(K_c a_c) - \Psi(a_c) + \ln \bar{\hat{\theta}}_c(e)\right)$$
$$\frac{dg}{da_c} = \frac{d\hat{\mathcal{L}}[q]}{da_c^2} = K_c \cdot \left(K_c \Psi'(K_c a_c) - \Psi'(a_c)\right),$$

we arrive at the following Newton-Raphson update rule:

$$\begin{aligned} a_c^{\text{new}} &= a_c - \frac{d\hat{\mathcal{L}}[q]}{da_c^2}^{-1} \frac{d\hat{\mathcal{L}}[q]}{da_c} \\ &= a_c - \frac{\Psi(K_c a_c) - \Psi(a_c) + \ln \bar{\hat{\theta}}_c(e)}{K_c \Psi'(K_c a_c) - \Psi'(a_c)}. \end{aligned}$$

However, since a_c is a Dirichlet parameter, $\hat{\mathcal{L}}[q]$ (and its gradient) is only defined for $a_c > 0$ which sometimes leads to a violation of the assumptions of Newton-Raphson, resulting in a negative value for a_c^{new}. One way around this is to simply re-initialize a_c whenever $a_c \leq 0$, but this is inelegant and slows convergence. Another approach is to introduce a change of variables in the gradient so that it remains well defined and then search for the root of this new equation. Here we choose $a_c = \exp(a'_c)$ and the derivative of the gradient with respect to a'_c becomes

$$\frac{dg}{da'_c} = \frac{dg}{da_c}\frac{da_c}{da'_c} = K_c \cdot \exp(a'_c) \cdot \left(K_c \Psi'(K_c \cdot \exp(a'_c)) - \Psi'(\exp(a'_c)\right),$$

leading to a new fixed point equation:

$$a_c^{\text{new}} = a_c \cdot \exp\left(-\frac{\Psi(K_c a_c) - \Psi(a_c) + \ln \bar{\hat{\theta}}_c(e)}{a_c \cdot \left(K_c \Psi'(K_c a_c) - \Psi'(a_c)\right)}\right).$$

This update is well behaved as long as a_A is initialized to some valid parameter setting (i.e., $a_c > 0$). (Convergence is still slow for roots where a_c is close to 0, but we have at least sidestepped the need to re-initialize.) It is identical to the MLE solution for $p(\theta_c|\alpha_c)$ where θ_c is observed except the variational parameter vector $\hat{\theta}_c$ is filling in for θ_c. In fact, any MLE solution could be adapted for our purposes and Minka (2000) describes several alternatives for computing the Dirichlet MLE.

The resultant coordinate ascent algorithm works just like any EM algorithm; the expected sufficient statistics ($\ln \hat{\theta}$ in this case) are computed during the E-step conditioned on some initial choice of parameter settings, and then these statistics are used

to compute a new MLE for the model parameters (α) in the M-step. By repeatedly alternating between these two steps we eventually converge to a local optimum. The only difference from standard EM is that the E-step is a full run to convergence of the VB algorithm outlined in Section 5.4, and the MLE computed in the M-step is approximate, based on $\hat{\mathcal{L}}[q]$ rather than the true likelihood which is intractable.

5.7 Conclusion

The algorithms outline in this chapter for training multi-weighted grammars is both simple to implement and highly general. In the end it is very similar to the standard EM algorithm for estimating rule weights in a PCFG. In fact, Equations 5.10 and 5.14 have the same form as the mean field VB algorithm for PCFGs (Kurihara and Sato, 2006). The biggest difference is that the multinomial parameters θ_c are defined, not by rule left-hand sides but rather by the more flexible μ function for multi-weighted grammar, which, in terms of implementation, is mainly a matter of indexing. Indeed, the algorithm can be used to train PCFGs or ordinary probabilistic synchronous grammars when the μ and ω functions are defined appropriately. It works just as well for end-to-end training of cascades of synchronous grammars, similar to the algorithm of Chiang et al. (2010) for finite state automata, provided the grammars are composable, but is more general still since it does not require that the multi-weighted grammar be decomposable into a cascade.

Chapter 6

Semantic Parsing

This chapter presents an adaptation of work previously published in Jones et al. (2012b). However, we have translated the tree-to-string model described there to the multi-weighted synchronous tree-string grammar framework laid out in Chapters 3 through 5.

In this chapter, we introduce a model for semantic parsing implemented in our multi-weighted probabilistic synchronous HRG framework described in Chapter 3. Although, as mentioned in Chapter 2, the term semantic parsing has been used to refer to a range of different tasks, we restrict consideration to a fairly typical special case where a system is trained on pairs of natural language sentences and their meaning representation expressions, as in figure 6.1(a), and the system must generalize to novel sentences. This observed meaning-sentence pair training condition is probably the most thoroughly studied task within semantic parsing, making it a good standard task to demonstrate the effectiveness of the framework since there are many other models with which to compare. Furthermore, the task lends itself to a fairly concise model description, making both testing and elaboration easier. In fact, the primary motivation of starting with this task is that it will be relatively straightforward to extend to our more general word learning scenario described in Chapter 8, and, importantly, most other approaches rely on alternative sources of supervision, such as database query answers or other lexical information not typically available to human learners Goldwasser et al. (2011); Liang et al. (2011). In fact, the model presented here corresponds very closely to a sub-module of the full word learner in Chapter 8.

Most semantic parsing models rely on an assumption of structural similarity between meaning representation and sentence. Since strict isomorphism is overly re-

(a)

Sentence: ‘what is the population of portland maine’
Meaning: answer(population(cityid(portland, maine)))

(b)

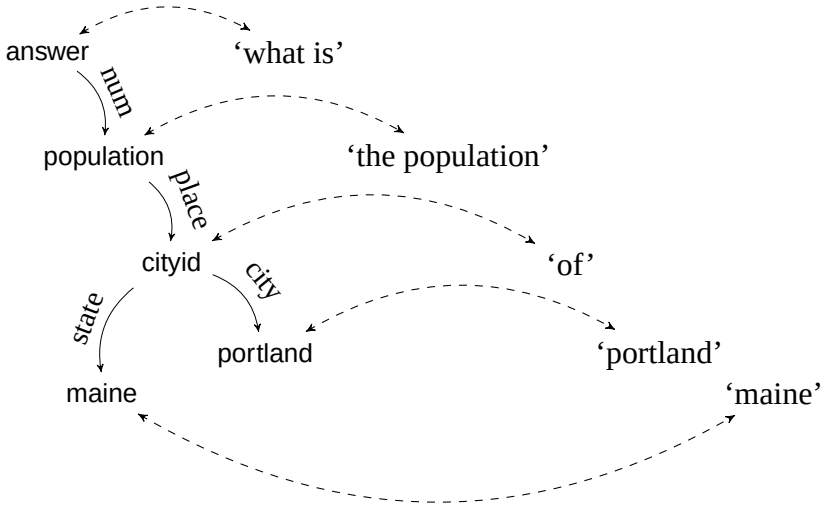

Figure 6.1: (a) An example sentence/meaning pair and (b) a possible mapping between them.

strictive, this assumption is often relaxed by applying transformations. Several approaches assume a tree structure to the sentence, meaning representation, or both (Ge and Mooney, 2005; Kate and Mooney, 2006; Wong and Mooney, 2006; Lu et al., 2008; Börschinger et al., 2011), and often involve tree transformations either between two trees or a tree and a string.

The synchronous grammar is well suited to formalizing such tree relations by jointly deriving both structures simultaneously. Yet, while many semantic parsing systems resemble the formalism, most have been proposed as standalone formalisms requiring custom algorithms, leaving it unclear how developments in one line of inquiry relate to others. We argue for a unifying theory of tree transformation based semantic parsing by presenting a synchronous grammar model built with the framework presented in Chapters 3 through 5 and draw connections to other similar systems.

Semantic parser training can be seen as a special case of word learning where the learner has perfect knowledge of the speaker’s intended meaning for every utterance. There is still referential ambiguity since the learner does not know which words con-

tribute which elements of the meaning, but the problem is much easier than one might expect when the learner must simultaneously infer both the speaker's intention and the meanings of the individual words. The grammar-based semantic parser presented here lays the groundwork for the word learning model in Chapter 8. With standard parsing and inference algorithms, grammar-based models are easily extended and adapted since one merely need add or change rules, freeing the modeler from inventing and implementing a custom algorithm for every model variation as is often necessary for other less standardized model families.

Seeing the semantic parser as a special case of the word learner, we can test and validate the learning performance of the model on a benchmark semantic parsing data set and demonstrate another dimension of the functionality of the word learning model. In order to calibrate the system we test it against other state-of-the-art systems on the most common data set, GeoQuery (Wong and Mooney, 2006).

6.1 Meaning representations and trees

In semantic parsing, a meaning representation is typically an expression in an application-specific machine interpretable language (e.g., a database query language like SQL). Typically, such languages are unambiguous context-free languages. Consequently each expression can be identified with a single tree without loss of information because the parse itself is a tree. Thus, these expression, either because they themselves are trees or because their parses are, fit neatly into the tree-based synchronous grammar framework outlined in Chapters 3-5.

The particular model we describe in this chapter is generally applicable to any application with such an unambiguous target meaning representation language, but our examples are drawn from a standard semantic parsing corpus, GeoQuery (Wong and Mooney, 2006). Figure 6.1(a) illustrates a typical sentence-meaning representation pair. The corpus centers on the task of learning a natural language interface for a database of geographical facts, and the data consists of a set of user questions expressed in natural language paired with a database query that would retrieve the answer to the question. The meaning representation database query is expressed in an unambiguous functional language where the nesting of the expression makes the tree shape easy to identify even by eye. The left-hand side of part (b) of Figure 6.1 shows the tree that corresponds to the meaning representation of the example in part (a). Functions and constants correspond to nodes of the tree and each function's arguments are identified

by its child edges.

The tree embodied by the basic nesting structure of the expression is elaborated slightly in the figure with edge labels identifying the argument type (i.e., the return type of the child function). This information is readily available from the grammar of the meaning representation language, a sample of which is shown below.

$$\begin{aligned} \text{NUM} &\rightarrow \text{population}(\text{PLACE}) \\ \text{PLACE} &\rightarrow \text{cityid}(\text{CITY}, \text{STATE}) \\ \text{CITY} &\rightarrow \text{portland} \\ \text{STATE} &\rightarrow \text{maine} \end{aligned}$$

The tree in the figure can be directly extracted from the parse tree. For consistency with the conventions for representing edge-labeled graphs and trees described in Chapter 7, we can translate the meaning representation in Figure 6.1(a) into the following predicate calculus expression:

$$\begin{aligned} \text{answer}(a) \wedge \text{num}(a,p) \wedge \text{population}(p) \\ \wedge \text{place}(p,c) \wedge \text{cityid}(c) \\ \wedge \text{city}(c,x) \wedge \text{portland}(x) \\ \wedge \text{state}(c,y) \wedge \text{maine}(y) \end{aligned}$$

Using this predicate calculus expression will make it easier to follow the example when we describe the model in terms of grammar rules.

6.2 Model

Our probabilistic model can be thought of as a translation model that first generates an expression in the source language (the meaning representation) and then generates its corresponding translation into natural language. First the meaning is generated according to the meaning representation grammar, guaranteeing that it is a well formed query. The model then generates the words corresponding to the tree, node by node, in a manner similar to the alignment illustrated in Figure 6.1(b). In terms of implementation, the scheme is realized as a synchronous grammar which, like the hybrid tree semantic parser (Lu et al., 2008) and WASP (Wong and Mooney, 2006), another synchronous grammar-based system, jointly generates the input meaning representation tree and the corresponding natural language string. The meaning representation is

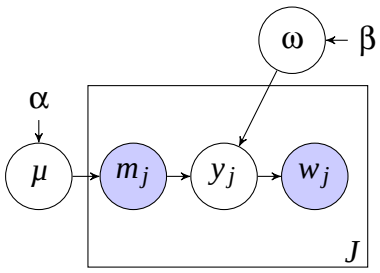

Figure 6.2: The generative model of the semantic parser over a corpus of J meaning-utterance pairs. w_j is an utterance, m_j is its corresponding meaning representation, and y_j is the latent mapping between them governed by the probabilistic synchronous grammar. μ is the set of multinomial parameters for the language model over meaning representations m, ω is the set of parameters for the utterances w given their corresponding meaning representation, and α and β are the parameters for their respective Dirichlet priors.

built up one production at a time according to a tree grammar while similar CFG-like productions are applied to the natural language side in lock-step formation, repeated until both the meaning representation and natural language are fully generated. In each step, the model selects a meaning representation rule and then builds the corresponding natural language by first choosing a word ordering pattern and then filling out that pattern with words drawn from a unigram distribution.

Figure 6.2 illustrates the graphical model, which generates the meaning-sentence pairs $\langle m_j, w_j \rangle$ of a corpus consisting of a total of J such pairs. First, a meaning representation m_j is drawn from a product of Dirichlet-multinomials defined with multinomial parameters μ and Dirichlet parameters α. Then a mapping y_j from meaning to words is generated according to conditional probability $P(w_j, y_j | m_j, \omega)$, which in turn is another product of Dirichlet-multinomials with multinomial and Dirichlet parameters ω and β, respectively.

The entire model can be implemented as a single synchronous grammar using the multi-weighted probabilistic extension defined in Section 3.4. Each rule consists of two monolingual components, one that generates the meaning representation tree according to an unordered tree grammar of the form described in Chapter 4, while the second monolingual component generates the words of the sentence as an ordinary CFG. Coupled together, the joint derivation simultaneously describes both how

the meaning is generated and how this meaning relates to the utterance, where the meaning-to-word map can be broken down into two types of choices: (1) word order decisions, and (2) word selection decisions. These mapping choices are probabilistically decided step by step, in synchrony with each step of meaning generation. Figure 6.1, for instance, illustrates which word choices might be made as each function is generated in the meaning. Additionally, ordering choices are also being made at each step, where the words corresponding to the function *population* are generated after those for *answer* and, similarly, a sequential ordering is chosen for the words "portland" and "maine". These decisions are encoded in the mapping variable y_j, which consists of the portion of the synchronous derivation that pertains only these two types of choices.

Thus, the entire probabilistic model can be summarized in the following formula:

$$P(m,x,w,\mu,\omega|\alpha,\beta) = P(\mu|\alpha)P(\omega|\beta)\prod_{j\in J} P(m_j|\mu)P(w_j,y_j|m_j,\omega)$$

where $P(\mu|\alpha)$ and $P(\omega|\beta)$ are products of Dirichlet probabilities, and $P(m_j|\mu)$ and $P(w_j,y_j|m_j,\omega)$ are product of multinomial distributions defined by the grammar which are defined in detail in the next two sections. The weights of the synchronous grammar are defined in a modular fashion according to a multi-weighted probabilistic grammar. One module defines $P(m_j|\mu)$, the other $P(w_j,y_j|m_j,\omega)$, where we use function $\mu(r)$ to denote the product of the weights on rule r that pertain to the probability of the meaning representation, and $\omega(r)$ to denote the product of the weights that govern the generation of the mapping from meaning to words. Thus, if x_j is a derivation of the synchronous grammar yielding m_j, w_j, and y_j, we can write

$$P(m_j|\mu) = \prod_{r\in x_j} \mu(r)$$
$$P(w_j,y_j|m_j,\omega) = \prod_{r\in x_j} \omega(r).$$

The μ weights govern the generation of the meaning representation and the ω weights the words of the utterance. In general, a single rule simultaneously contributes to the generation of both the meaning and words, implying that the same rule can have both a μ and ω weight, something that, while impossible in a conventional weighted grammar, is perfectly legal using a multi-weighted grammar.

6.2.1 Meaning generation: $P(m_j|\mu)$

The model starts by generating the function corresponding to the root node of the tree and then generates its children, recursively generating each function until the meaning representation is complete, where the probability of each function is conditioned on information about its parent. Specifically, each function is conditioned on the label and list of argument types of its parent, a combination which we will refer to as the function's *signature*. Similarly, at the same time as a function is generated, the model simultaneously selects its signature. For instance, in generating the node labeled *cityid* and its two children in the tree in Figure 6.1, the model first chooses function signature *cityid/CITY/STATE* conditioned on the signature of its parent *population/PLACE* and then proceeds to choose the signatures of *cityid*'s first and second arguments *portland* and *maine*, where we treat constants as functions with no arguments so their signatures are just *portland* and *maine*, respectively.

This language model over meaning representation expressions can be defined formally as

$$P(m|\mu) = \prod_{f \in m} P(sig(f)|sig(parent(f)), arg(f), \mu)$$

where $sig(f)$ is the signature of function f, $parent(f)$ is its parent, and $arg(f)$ is the index into the argument list of f's parent.

This process can be captured by an unordered tree grammar of the type described in Chapter 4 that alternates between node (i.e., functions label) rules and edge (i.e., argument type) rules. In keeping with that grammar definition, we define two sets of nonterminals, node-generating and edge-generating nonterminals, and index each by the combination of function signature and the argument number to be generated. That is, there is a node-generating nonterminal specifically for choosing the CITY child of function *cityid*, and another for choosing its STATE child. Notationally, we identify such a node-generating nonterminal by N[CITYID/CITY/STATE] for the CITY child and N[CITYID/CITY/STATE] for the STATE child. Similarly, the corresponding edge-generating nonterminals are denoted by E[CITYID/CITY/STATE] and E[CITYID/CITY/STATE].

Table 6.1 presents the fragment of the grammar that generates the meaning representation in Figure 6.1. Rules m1 through m5 are node-generating rules which determine the function label and the number of its arguments. The weights of these rules are defined such that they are equivalent to the conditional probability of the

$$\text{N[START]} \rightarrow \text{answer}(x) \wedge \text{E[ANS/}\underline{\text{NUM}}\text{]}(x) \quad \text{(m1)}$$

$$\text{N[ANS/}\underline{\text{NUM}}\text{]} \rightarrow \lambda x.\text{population}(x) \wedge \text{E[POP/}\underline{\text{PLACE}}\text{]}(x) \quad \text{(m2)}$$

$$\text{N[POP/}\underline{\text{PLACE}}\text{]} \rightarrow \lambda x.\text{cityid}(x) \wedge \text{E[CITYID/}\underline{\text{CITY}}\text{/STATE]}(x) \wedge \text{E[CITYID/CITY/}\underline{\text{STATE}}\text{]}(x) \quad \text{(m3)}$$

$$\text{N[CITYID/}\underline{\text{CITY}}\text{/STATE]} \rightarrow \lambda x.\text{portland}(x) \quad \text{(m4)}$$

$$\text{N[CITYID/CITY/}\underline{\text{STATE}}\text{]} \rightarrow \lambda x.\text{maine}(x) \quad \text{(m5)}$$

$$\text{E[ANS/}\underline{\text{NUM}}\text{]} \rightarrow \lambda x.\text{num}(x,y) \wedge \text{N[ANS/}\underline{\text{NUM}}\text{]}(y) \quad \text{(m11)}$$

$$\text{E[POP/}\underline{\text{PLACE}}\text{]} \rightarrow \lambda x.\text{place}(x,y) \wedge \text{N[POP/}\underline{\text{PLACE}}(y) \quad \text{(m12)}$$

$$\text{E[CITYID/}\underline{\text{CITY}}\text{/STATE]} \rightarrow \lambda x.\text{city}(x,y) \wedge \text{N[CITYID/}\underline{\text{CITY}}\text{/STATE]}(y) \quad \text{(m13a)}$$

$$\text{E[CITYID/CITY/}\underline{\text{STATE}}\text{]} \rightarrow \lambda x.\text{state}(x,y) \wedge \text{N[CITYID/CITY/}\underline{\text{STATE}}\text{]}(y) \quad \text{(m13b)}$$

Table 6.1: A grammar fragment that generates the meaning representation in Figure 6.1. The top rules are node-generating, which select the function's label and arity, while the bottom edge-generating rules produce the argument type labels.

function signature identified with the nonterminal(s) on the right given the function signature associated with the nonterminal on the left hand side of the rule. For example, we assign a weight equivalent to $P(pop/\text{PLACE} \mid ans/\underline{\text{NUM}})$ to rule m2 and $P(portland \mid cityid/\underline{\text{CITY}}/\text{STATE})$ to rule m4. Thus, these rules model the probability of choosing a particular function given the signature of its parent.

The edge-generating nonterminals are expanded according to rules m11 through m13b. These rules simply produce the edge label corresponding to the specified type and hand off generation to the next node-generating nonterminal which then proceeds to perform the next function selection. These edge-generating steps are perfectly deterministic, so each has a probability of one.

Figure 6.3(a) shows the derivation tree of the meaning representation in Figure 6.1 under the grammar in Table 6.1. Letting $\mu(r)$ be the product of the meaning representation language model weights of rule r, then the probability of this derivation (and the

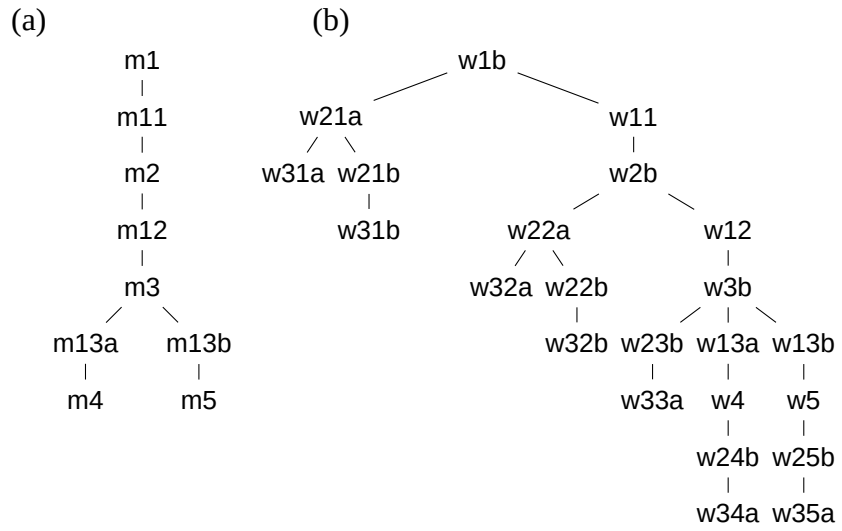

Figure 6.3: A derivation tree for (a) the meaning representation and (b) the meaning-sentence mapping illustrated in Figure 6.1 with the monolingual meaning grammar (Table 6.1) and synchronous meaning-sentence grammar (Table 6.2), respectively. Tracking the correspondence between the rules of the meaning representation grammra and the meaning-word grammar, the meaning representation can be seen as the skeleton of the meaning-to-word map.

example meaning representation) would be

$$\begin{aligned}
P(m|\mu) &= \mu(m1)\mu(m11)\mu(m2)\mu(m12)\mu(m3)\mu(m13a)\mu(m4)\mu(m13b)\mu(m5) \\
&= \mu(m1)\mu(m2)\mu(m3)\mu(m4)\mu(m5) \\
&= P(ans/\text{NUM}\,|\text{START},\mu)\cdot P(pop/\text{PLACE}\mid ans/\underline{\text{NUM}},\mu) \\
&\quad \cdot P(cityid/\text{CITY}/\text{STATE}|pop/\underline{\text{PLACE}},\mu) \\
&\quad \cdot P(portland|cityid/\underline{\text{CITY}}/\text{STATE},\mu) \\
&\quad \cdot P(maine\mid cityid/\text{CITY}/\underline{\text{STATE}},\mu)
\end{aligned}$$

To see how this relates to the formal definition of multi-weighted grammars in Section 3.4, observe that the weights of rules m1-m5 can be implemented with just two feature functions: $lhs(r)$ that returns the left-hand side nonterminal of r, and $sig(r)$ that returns the name of the unary and the type of it's argument encoded in the nonterminal on the right-hand side. Given these feature functions, a factorization function of $\varphi(r) = \langle lhs(r), sig(r)\rangle$ yields the probabilities above. As for rules m11-m13b, they all have a

μ value of 1, which could be implemented with a $\varphi(r) = \langle \emptyset, \emptyset \rangle$ that simply returns a dummy value with a weight of 1.

6.2.2 Sentence generation: $P(w_j, y_j | m_j, \omega)$

Sentence generation can be conceptualized as a process of translating meaning to words through a sequence of permutation and word substitution/insertion operations. Walking down the meaning representation tree node by node, the model first chooses a particular linearization of the node and its children and then inserts words into that linearization, generated according to a unigram distribution. As with the meaning representation, both the choice of linearization and word selection are conditioned on the full signature of the corresponding function. A linearization of node f consists of a particular ordering among the children and a choice of word insertion points, where words can be optionally inserted before, after, or anywhere in between children. For example, there are three possible orderings for the *population* node and its children.

CHILD

WORDS CHILD

CHILD WORDS

WORDS CHILD WORDS

Similarly, there are 16 possible linearizations for *cityid* (2 different permutations of the children with 8 different choices of word insertion points for each). In general, for a function of arity k, there are $k!$ permutations of its children and $k+1$ locations in these sequences where words can be inserted, leading to $k! \cdot 2^{k+1}$ possible linearizations to choose from. We force a single linearization (i.e., WORDS) for constants so that they each contribute some sequence of words within the sentence.

To define the probability distribution formally, let latent variable y be the sequence of permutation and substitution operations for the entire sentence, and y_f be a linearization followed by a sequence of word substitution operations that translate a particular function f with arity k in the meaning representation into a substring of the sentence. Let ℓ_f be the linearization containing $k+1$ places for words, and $w_{f,1}, \ldots, w_{f,k+1}$ be the

particular word sequences chosen to fill those place holders.

$$P(w,y|m,\omega) = \prod_{f\in m} P(y_f = \langle \ell_f, w_{f,1}, ..., w_{f,k+1}\rangle | sig(f),\omega)$$
$$= P(\ell_f|sig(f),\omega)\prod_{i=1}^{k+1} P(w_{f,i}|sig(f),\omega)$$

The probability $P(\ell_f|sig(f),\omega)$ is just a multinomial distribution over linearizations and the word substitution $w_{f,i}$ is defined according to a unigram distribution associated with $sig(f)$, where $w_{f,i}$ is a sequence of words $word_{i,1}...word_{i,n_i}$:

$$P(w_{f,i}|sig(f),\omega) = \prod_{j=1}^{n-1} P(word_{i,j}|sig(f),\omega)P(continue|sig(f),\omega)$$
$$\cdot P(word_{i,n_i}|sig(f),\omega)P(stop|sig(f),\omega).$$

In terms of implementation, these linearization and word insertion operations can be specified by extending each production of the monolingual meaning-generating grammar in Table 6.1 with CFG-style string-generating productions, one for each possible linearization. Words can then be generated according to string-only monolingual rules that expand word-generating nonterminals, denoted by $W[sig(f)]$ and $U[sig(f)]$. Table 6.2 lists the rules necessary for producing the mapping in Figure 6.1(b), and Figure 6.3(b) illustrates the derivation tree. The four possible linearizations for *population* corresponding to rule m2 would lead to the following set of synchronous rules:

$$\text{N[ANS/}\underline{\text{NUM}}\text{]} \rightarrow \langle\ \lambda x.\text{population}(x) \wedge \text{E[POP/}\underline{\text{PLACE}}\text{]}_{\boxed{1}}(x)\ \|$$
$$\ell_{population}\ \rangle$$

where $\ell_{population}$ is one of

E[POP/$\underline{\text{PLACE}}$]$_{\boxed{1}}$ (w2a)

W[POP/PLACE] E[POP/$\underline{\text{PLACE}}$]$_{\boxed{1}}$ (w2b)

E[POP/$\underline{\text{PLACE}}$]$_{\boxed{1}}$ W[POP/PLACE] (w2c)

W[POP/PLACE] E[POP/$\underline{\text{PLACE}}$]$_{\boxed{1}}$ W[POP/PLACE]. (w2d)

The particular mapping in Figure 6.1 linearizes *population* by inserting words before the substring corresponding to the translation of its single subtree as dictated by rule w2b.

$$\text{N[START]} \rightarrow \langle\ \text{answer}(x) \wedge \text{E[ANS/}\underline{\text{NUM}}\text{]}_{\boxed{1}}(x)\ \|\ \text{W[ANS/NUM] E[ANS/}\underline{\text{NUM}}\text{]}_{\boxed{1}}\ \rangle \quad \text{(w1b)}$$

$$\text{N[ANS/}\underline{\text{NUM}}\text{]} \rightarrow \langle\ \lambda x.\text{population}(x) \wedge \text{E[POP/}\underline{\text{PLACE}}\text{]}_{\boxed{1}}(x)\ \|\ \text{W[POP/PLACE] E[POP/}\underline{\text{PLACE}}\text{]}_{\boxed{1}}\ \rangle \quad \text{(w2b)}$$

$$\text{N[POP/}\underline{\text{PLACE}}\text{]} \rightarrow \langle\ \lambda x.\text{cityid}(x) \wedge \text{E[CITYID/}\underline{\text{CITY}}\text{/STATE]}_{\boxed{1}}(x) \wedge \text{E[CITYID/CITY/}\underline{\text{STATE}}\text{]}_{\boxed{2}}(x)\ \|\ \text{W[CITYID/CITY/STATE]}\ \text{E[CITYID/}\underline{\text{CITY}}\text{/STATE]}_{\boxed{1}}\ \text{E[CITYID/CITY/}\underline{\text{STATE}}\text{]}_{\boxed{2}}\ \rangle \quad \text{(w3b)}$$

$$\text{N[CITYID/}\underline{\text{CITY}}\text{/STATE]} \rightarrow \langle\ \lambda x.\text{portland}(x)\ \|\ \text{W[PORTLAND]}\ \rangle \quad \text{(w4)}$$

$$\text{N[CITYID/CITY/}\underline{\text{STATE}}\text{]} \rightarrow \langle\ \lambda x.\text{maine}(x)\ \|\ \text{W[MAINE]}\ \rangle \quad \text{(w5)}$$

$$\text{E[ANS/}\underline{\text{NUM}}\text{]} \rightarrow \langle\ \lambda x.\text{num}(x,y) \wedge \text{N[ANS/}\underline{\text{NUM}}\text{]}_{\boxed{1}}(y)\ \|\ \text{N[ANS/}\underline{\text{NUM}}\text{]}_{\boxed{1}}\ \rangle \quad \text{(w11)}$$

$$\text{E[POP/}\underline{\text{PLACE}}\text{]} \rightarrow \langle\ \lambda x.\text{place}(x,y) \wedge \text{N[POP/}\underline{\text{PLACE}}_{\boxed{1}}(y)\ \|\ \text{N[POP/}\underline{\text{PLACE}}_{\boxed{1}}\ \rangle \quad \text{(w12)}$$

$$\text{E[CITYID/}\underline{\text{CITY}}\text{/STATE]} \rightarrow \langle\ \lambda x.\text{city}(x,y) \wedge \text{N[CITYID/}\underline{\text{CITY}}\text{/STATE]}_{\boxed{1}}(y)\ \|\ \text{N[CITYID/}\underline{\text{CITY}}\text{/STATE]}_{\boxed{1}}\ \rangle \quad \text{(w13a)}$$

$$\text{E[CITYID/CITY/}\underline{\text{STATE}}\text{]} \rightarrow \langle\ \lambda x.\text{state}(x,y) \wedge \text{N[CITYID/CITY/}\underline{\text{STATE}}\text{]}_{\boxed{1}}(y)\ \|\ \text{N[CITYID/CITY/}\underline{\text{STATE}}\text{]}_{\boxed{1}}\ \rangle. \quad \text{(w13b)}$$

$$\text{W[}\mathit{SIG}\text{]} \rightarrow \langle\ -\ \|\ \text{U[}\mathit{SIG}\text{] W[}\mathit{SIG}\text{]}\ \rangle \quad \text{(w21-25a)}$$

$$\text{W[}\mathit{SIG}\text{]} \rightarrow \langle\ -\ \|\ \text{U[}\mathit{SIG}\text{]}\ \rangle \quad \text{(w21-25b)}$$

$$\text{U[ANS/NUM]} \rightarrow \langle\ -\ \|\ \text{what}\ \rangle \quad \text{(w31a)}$$

$$\text{U[ANS/NUM]} \rightarrow \langle\ -\ \|\ \text{is}\ \rangle \quad \text{(w31b)}$$

$$\text{U[POP/PLACE]} \rightarrow \langle\ -\ \|\ \text{the}\ \rangle \quad \text{(w32a)}$$

$$\text{U[POP/PLACE]} \rightarrow \langle\ -\ \|\ \text{population}\ \rangle \quad \text{(w32b)}$$

$$\text{U[CITYID/CITY/STATE]} \rightarrow \langle\ -\ \|\ \text{of}\ \rangle \quad \text{(w33a)}$$

$$\text{U[PORTLAND]} \rightarrow \langle\ -\ \|\ \text{portland}\ \rangle \quad \text{(w34a)}$$

$$\text{U[MAINE]} \rightarrow \langle\ -\ \|\ \text{maine}\ \rangle \quad \text{(w35a)}$$

Table 6.2: A grammar for the meaning-to-sentence map in Figure 6.1: (from top) function generating/linearization, edge generating, and word stopping and generating rules.

The meaning-to-word mapping weights for each of these synchronous rules, denoted by $\omega(r)$, is simply the weight of the particular linearization. So, for example, the word-generating weight of rule w2b is constructed to be equivalent to

$$\omega(w2b) = P(\text{WORDS CHILD} \mid pop/\text{PLACE}, \omega).$$

The word generation unigram is implemented by two different kinds of rules, one kind of rule for determining when the unigram stops generating words, and one for actually generating the specific words. Specifically, there are a pair of stopping rules following the pattern of w21-25a and w21-25b in Table 6.2 for each function signature, which are weighted according to $P(stop|sig(f))$ and $P(continue|sig(f))$. For instance, there are two such rules for *pop/PLACE*,

$$\text{W[POP/PLACE]} \rightarrow \langle\ - \parallel \text{U[POP/PLACE] W[POP/PLACE]}\ \rangle \quad \text{(w22a)}$$

$$\text{W[POP/PLACE]} \rightarrow \langle\ - \parallel \text{U[POP/PLACE]}\ \rangle. \quad \text{(w22b)}$$

Rule w22a is assigned a weight of $P(continue|\ pop/\text{PLACE})$ and rule w22b a weight of $P(stop|\ pop/\text{PLACE})$. A chain of such rules in a derivation simulates the flipping of a coin after each word is generated to determine whether to stop or continue adding more words to the string. Similar rules necessary for each function signature in the corpus

$$\text{W[ANS/NUM]} \rightarrow \langle\ - \parallel \text{U[ANS/NUM] W[ANS/NUM]}\ \rangle \quad \text{(w21a)}$$

$$\text{W[ANS/NUM]} \rightarrow \langle\ - \parallel \text{U[ANS/NUM]}\ \rangle \quad \text{(w21b)}$$

$$\text{W[CITYID/CITY/STATE]} \rightarrow \langle\ - \parallel \text{U[CITYID/CITY/STATE] W[CITYID/CITY/STATE]}\ \rangle \quad \text{(w23a)}$$

$$\text{W[CITYID/CITY/STATE]} \rightarrow \langle\ - \parallel \text{U[CITYID/CITY/STATE]}\ \rangle \quad \text{(w23b)}$$

$$\text{W[PORTLAND]} \rightarrow \langle\ - \parallel \text{U[PORTLAND] W[PORTLAND]}\ \rangle \quad \text{(w24a)}$$

$$\text{W[PORTLAND]} \rightarrow \langle\ - \parallel \text{U[PORTLAND]}\ \rangle \quad \text{(w24b)}$$

$$\text{W[MAINE]} \rightarrow \langle\ - \parallel \text{U[MAINE] W[MAINE]}\ \rangle \quad \text{(w25a)}$$

$$\text{W[MAINE]} \rightarrow \langle\ - \parallel \text{U[MAINE]}\ \rangle \quad \text{(w25b)}$$

The words themselves are generated according to rules such as

$$\text{U[POP/PLACE]} \rightarrow \langle\ - \ \|\ \text{the}\ \rangle \tag{w32a}$$
$$\text{U[POP/PLACE]} \rightarrow \langle\ - \ \|\ \text{population}\ \rangle \tag{w32b}$$
$$\text{U[POP/PLACE]} \rightarrow \langle\ - \ \|\ \text{what}\ \rangle \tag{w32c}$$
$$\text{U[POP/PLACE]} \rightarrow \langle\ - \ \|\ \text{is}\ \rangle \tag{w32d}$$
$$\vdots$$

These word-generating rules are each weighted according to the unigram probability so that, for instance, rule w32c receives weight equal to $P(\text{what} \mid pop/\text{PLACE})$ and rule w32d receives weight $P(\text{is} \mid pop/\text{PLACE})$. In general, every function can map to any sequence of words in the vocabulary of the language, requiring rules such as w31a through w35a for every word and function in the corpus vocabulary. [1]

The deterministic edge-generating rules of the monolingual meaning representation grammar are extended as per rules w11 to w13b to simply propagate the nonterminals through the string generating portion of derivations. Propagating the node-generating nonterminals into the string side ensures that the strings corresponding to the translation of the subtrees of the meaning representation are properly inserted into the string. Again, these expansions are deterministic.

To compute the probability of the mapping conditioned on the meaning representation, again, we simply multiply the ω weights for each rule in the derivation.

$$\begin{aligned}
P(w,y|m,\omega) &= \omega(w1b)\omega(w21a)\omega(w31a)\omega(w21b)\omega(w31b) \\
&\quad \cdot \omega(w11)\omega(w2b)\omega(w22a)\omega(w32a)\omega(w22b)\omega(r32b) \\
&\quad \cdot \omega(w12)\omega(w3b)\omega(w23b)\omega(w33a) \\
&\quad \cdot \omega(w13a)\omega(w4)\omega(w24b)\omega(w34a) \\
&\quad \cdot \omega(w13b)\omega(w5)\omega(w25b)\omega(w35a) \\
&= \omega(w1b)\omega(w21a)\omega(w31a)\omega(w21b)\omega(w31b) \\
&\quad \cdot \omega(w2b)\omega(w22a)\omega(r32a)\omega(w22b)\omega(r32b) \\
&\quad \cdot \omega(w3b)\omega(w23b)\omega(w33a) \\
&\quad \cdot \omega(w4)\omega(w24b)\omega(w34a)\omega(w5)\omega(w25b)\omega(w35a)
\end{aligned}$$

[1] There are roughly 25,000 rules in the transducers in our experiments, and the majority of these implement the unigram word distributions since every entity in the MR may potentially produce any of the words it is paired with in training.

This derivation weight, when interpreted in terms of probabilities resolves to

$$\begin{aligned}
P(w,y|m,\omega) = {} & P(\text{WORDS CHILD}|\mathit{ans}/\text{NUM},\omega) \\
& \cdot P(\text{`what'}|\mathit{ans}/\text{NUM},\omega)P(\mathit{continue}|\mathit{ans}/\text{NUM},\omega) \\
& \cdot P(\text{`is'}|\mathit{ans}/\text{NUM},\omega)P(\mathit{stop}|\mathit{ans}/\text{NUM},\omega) \\
& \cdot P(\text{WORDS CHILD}|\mathit{population}/\text{PLACE},\omega) \\
& \cdot P(\text{`the'}|\mathit{population}/\text{PLACE},\omega)P(\mathit{continue}|\mathit{population}/\text{PLACE},\omega) \\
& \cdot P(\text{`population'}|\mathit{population}/\text{PLACE},\omega) \\
& \cdot P(\mathit{stop}|\mathit{population}/\text{PLACE},\omega) \\
& \cdot P(\text{WORDS CHILD}_1\ \text{CHILD}_2|\mathit{cityid}/\text{CITY}/\text{STATE},\omega) \\
& \cdot P(\text{`of'}|\mathit{cityid}/\text{CITY}/\text{STATE})P(\mathit{stop}|\mathit{cityid}/\text{CITY}/\text{STATE},\omega) \\
& \cdot P(\text{WORDS}|\mathit{portland},\omega)P(\text{`portland'}|\mathit{portland},\omega)P(\mathit{stop}|\mathit{portland},\omega) \\
& \cdot P(\text{WORDS}|\mathit{maine},\omega)P(\text{`maine'}|\mathit{maine},\omega)P(\mathit{stop}|\mathit{maine},\omega)
\end{aligned}$$

The derivation first chooses a linearization for *answer*, then inserts words "what" and "is", linearizes *population* and its arguments, inserts words "the" and "population", and so on until the entire meaning representation has been translated.

After carrying over the meaning representation weights $\mu(r)$ from the monolingual meaning representation grammar, we can compute $P(m|\mu)$ in exactly the same way as before, by walking the joint derivation and multiplying out the $\mu(r)$ weights. In fact, the monolingual derivation for the meaning representation shown in Figure 6.3(a) is embedded within the synchronous derivation shown in part (b) of the figure, it has simply been extended by adding on the linearization and unigram productions.

Defining the model formally requires just four feature functions: the signature function $sig(r)$ defined as before, $\ell(r)$ which identifies the word order linearization pattern, $word(r)$ which identifies the word on the right-hand side, and $stop(r)$, a boolean function returning true if and only if the rule is of the form w21-25b. Folding in the factors from the meaning generation model definition of the previous section, rules w1b-w5 have factorization functions of the form

$$\varphi(r) = \langle lhs(r), sig(r)\rangle \cdot \langle sig(r), \ell(r)\rangle.$$

The $\langle lhs(r), sig(r)\rangle$ factor is inherited from the definition of the μ weights, and the second factor corresponds to the linearization pattern probability. This results in a weight vector of the form

$$\mu(r)\cdot\omega(r) = P(sig(r)|lhs(r),\mu)\cdot P(\ell(r)|sig(r),\omega)$$

The word-generating portion of rules w11-w13b are deterministic, so the ω weights can be defined by using an empty factor $\langle \emptyset, \emptyset \rangle$ just as before in the definition of μ. Rules w21-25a and w35a all only generate words, contributing nothing to the meaning, so have no μ weights (or, equivalently, have a weight of 1 defined as per the empty factor). Rules w21-25a and w21-25b have a word-generating factor of the form $\langle lhs(r), stop(r) \rangle$, resulting in a weight of $\omega(r) = P(stop(r)|lhs(r), \omega)$. Finally, rules w31a-w35a have factors of the form $\langle lhs(r), word(r) \rangle$, with a weight of $\omega(r) = P(word(r)|lhs(r), \omega)$.

Because all feature pairs with μ weights are defined exclusively in terms of the meaning representation m, and all feature pairs with ω weights are defined with features from word portion of the rules on the right, the model is guaranteed to factorize as per $P(m, w|\mu, \omega) = P(m|\mu)P(w|m, \omega)$, as described by the plate diagram in Figure 6.2.

6.3 Relation to other models

The multi-weighted synchronous grammar model can be viewed either as a generative procedure for building up two separate structures or as a transformative machine that takes one as input and produces another as output (a la tree transducers). Different semantic parsing approaches have taken one or the other view, and both can be captured in this single framework. WASP (Wong and Mooney, 2006) is an example of the former perspective, coupling the generation of the meaning representation and sentence with a different sort of synchronous grammar. The most significant difference from our approach is that they use machine translation techniques for automatically extracting rules from parallel corpora (Galley et al., 2004). Our approach differs in that we specify general rules based on the *language* of meaning representations rather than the particular *examples* of meaning representations in the training corpus. A key advantage of this language-based approach over the example-based approach is that the mapping rules can be specified without assuming the meanings are observed. In the narrow context of semantic parsing where training is conducted with observed meanings this may seem like a subtle and purely theoretical distinction. In fact, since WASP only extracts the rules required to explain the alignments in the training examples, its grammars tend to be much smaller than ours, leading to a more efficient parser. However, abstraction away from reliance on observed meaning representations during training is crucial for generalization to the word leaning scenario in Chapter 8 since meanings are completely latent, rendering a WASP-like alignment-based approach untenable.

The hybrid tree model (Lu et al., 2008) takes an approach that is in some ways more similar to our model. In fact, there is a very close correspondence between the parameters of our model and those of the hybrid tree. Furthermore, like our model, the hybrid tree system does not require alignments between observed meaning representations and sentences for grammar extraction. However, they do not represent their model as an explicit grammar, instead inventing a new notion of the "hybrid tree" that unifies the meaning and sentence into a single structure requiring a custom parsing algorithm and additional work in disentangling the two at test time. Our synchronous grammar, on the other hand, naturally captures many of the same probabilistic dependencies while making use of a more standard grammar framework which builds upon a larger body of theory. This reliance on a general grammatical framework lends us greater flexibility when it comes to model extension and adaptation, something we make extensive use of in Chapter 8.

KRISP (Kate and Mooney, 2006) uses string classifiers to label substrings of the sentence with functions and constants from the meaning representation. To focus search, they impose an ordering constraint based on the structure of the meaning representation tree, which they relax by allowing the re-ordering of sibling nodes and devise a procedure for recovering the meaning from the permuted tree. This procedure corresponds to backward-application in synchronous grammars, identifying the most likely source tree given a particular target string. As with the hybrid tree, KRISP is not based on a grammar, even if it closely approximates one, which makes it harder to extend and forces the authors to rely on custom algorithms tailored specifically to their problem.

Börschinger et al. (2011) take a similar stance to ours but argue for the PCFG as an alternative model class, which they advocate on the basis that PCFGs facilitate the application of conventional grammar induction techniques. We are sympathetic to this argument, particularly since our framework is a generalization of PCFGs and benefits from the same features and allows us to incorporate the same modeling techniques. However, the PCFG is less amenable to conceptualizing correspondences between parallel structures, and their model is more restrictive, only applicable to domains with finite meaning representation languages, since their non-terminals encode entire meaning representations. The multi-weighted synchronous grammar framework, on the other hand, allows us to exploit the compositional properties of the meaning representations so that linearizations and word probabilities are dependent on local features (function signatures, specifically). Furthermore, the PCFG approach can quickly become conceptually unwieldy as it is extended into more complex models, something

that is much easier for us.

Finally, the UBL system of Kwiatkowski et al. (2010) also takes a grammar-based approach, employing a restricted context-free variant of CCG, making it somewhat similar to our framework. The biggest departure from our approach and that of any of the others mentioned is in how it treats the meaning representation. While all the other approaches essentially leave the meaning representation intact and utilize this to restrict the search space, Kwiatkowski et al. (2010) decomposes it into more arbitrary fragments as enumerated by the *split* function. This allows UBL to analyze the meaning representation in different ways which permits a larger theoretical space of potential meaning-to-word mappings. However, this flexibility comes at the cost of greater parsing complexity. In the experiments described in Section 6.4, for example, we run UBL as well as WASP and the hybrid tree and find that while it consistently takes 6-8 hours across four languages to train on a 600 training pair subset of GeoQuery, while the others all complete in under two hours. WASP takes even less than half an hour due to the relatively small number of rules in its grammar. This computational complexity that UBL must contend with makes it harder to scale to accommodate the additional complexity inherent in less constrained word learning settings.

6.4 Experiments

6.4.1 Evaluation

We evaluate the system on GeoQuery (Wong and Mooney, 2006), a parallel corpus of 880 English questions and database queries about United States geography, 250 of which were translated into Spanish, Japanese, and Turkish. We present here additional translations of the full 880 sentences into German, Greek, and Thai. For evaluation, following from Kwiatkowski et al. (2010), we reserve 280 sentences for test and train on the remaining 600. During development, we use cross-validation on the 600 sentence training set. At test, we run once on the remaining 280 and perform 10 fold cross-validation on the 250 sentence sets.

Training consists of parsing meaning-sentence pairs and using the resultant parse forests and the VB algorithm described in Chapter 5 to estimate rule weights. Thus, at test time, we parse just the sentence and use the translation grammar as described in Section 3.3.2 to find the meaning associated with the most probable joint meaning-sentence derivation. To judge correctness, we follow standard practice and submit each

parse as a GeoQuery database query, and say the parse is correct only if the answer matches the gold standard. We report raw accuracy (the percentage of sentences with correct answers), as well as F1: the harmonic mean of precision (the proportion of correct answers out of sentences with a parse) and recall (the proportion of correct answers out of all sentences).[2]

We run three other state-of-the-art systems for comparison. *WASP* (Wong and Mooney, 2006) and the *hybrid tree* (Lu et al., 2008) are chosen to represent tree transformation based approaches, and, while this comparison is our primary focus, we also report *UBL-S* (Kwiatkowski et al., 2010) as a non-tree based top-performing system. The hybrid tree is notable as the only other system based on a generative model, and *uni-hybrid*, a version that uses a unigram distribution over words, is very similar to our own model. We also report the best performing version, *re-hybrid*, which incorporates a discriminative re-ranking step.

We report the performance of our synchronous grammar under three different training conditions: *scfgEM* using EM, *scfgVB-auto* using VB with empirical Bayes, and *scfgVB-hand* using hyper-parameters manually tuned on the German training data with three different hyper-paramter settings, one for meaning generation parameters (α of 0.3), one for the different linearization patterns such as shown in the sentence-generating side of rules such as rules w1b-w5 in Table 6.2 (with a β of 0.8), and one for word generation rules (β of 0.25).

Table 6.3 shows results for 10 fold cross-validation on the training set. The results highlight the benefit of the Dirichlet prior, whether manually or automatically set. VB improves over EM considerably, most likely because (1) the handling of unknown words and meaning representation functions allows it to return an analysis for all sentences, and (2) the sparse Dirichlet prior favors fewer rules, reasonable in this setting where only a few words are likely to share the same meaning.

On the test set (Table 6.4), we only run the model variants that perform best on the training set. Test set accuracy is consistently higher for the VB trained synchronous grammar than the other tree transformation based models (and often highest overall), while f-score remains competitive.[3]

The relatively high performance of our model is likely due in large part to two factors owing to VB.[4] First, the sparse prior is a better match to our problem where

[2]Note that accuracy and f-score reduce to the same formula if there are no parse failures.

[3]Numbers differ slightly here from previously published results due to the fact that we have standardized the inputs to the different systems.

[4]Kwiatkowski et al. (2012) also observed that their incremental VB algorithm applied to a model

DEV	geo600 - 10 fold cross-val							
	German				Greek			
	Acc		F1		Acc		F1	
UBL-S	76.	7	76.	9	76.	2	76.	5
WASP	66.	3	75.	0	71.	2	79.	7
uni-hybrid	61.	7	66.	1	71.	0	75.	4
re-hybrid	62.	3	69.	5	70.	2	76.	8
scfgEM	61.	7	67.	9	67.	3	73.	2
scfgVB-auto	74.	0	74.	0	•**79**.	**8**	•**79**.	**8**
scfgVB-hand	•**78**.	**0**	•**78**.	**0**	79.	0	79.	0
	English				Thai			
UBL-S	**85**.	**3**	**85**.	**4**	74.	0	74.	1
WASP	73.	5	79.	4	69.	8	73.	9
uni-hybrid	76.	3	79.	0	71.	3	73.	7
re-hybrid	77.	0	82.	2	71.	7	76.	0
scfgEM	73.	5	78.	1	69.	8	72.	9
scfgVB-auto	81.	2	81.	2	74.	7	74.	7
scfgVB-hand	•83.	7	•83.	7	•**76**.	**7**	•**76**.	**7**

Table 6.3: Accuracy and F1 score comparisons on the geo600 training set. Highest scores are in bold, while the highest among the *tree based* models are marked with a bullet. The dotted line separates the tree based from non-tree based models.

only a few words are likely to be used to express a particular concept in the meaning representation language. Second, the prior also allows the model to generalize to words and meaning representation symbols previously unseen during training, acting as a kind smoothing scheme. WASP and the hybrid tree simply fail to return a parse in the cases of unknown words, negatively impacting recall, but the Bayesian prior permits our model to always propose a meaning representation for any given sentence. It is less clear why our approach improves over UBL-S, which employs a two-pass parsing approach to return a best guess in the case of failure, but perhaps the Dirichlet prior's ability to model sparsity helps here as well.

We have argued that tree transformation based semantic parsing can benefit from the literature on formal language theory and tree automata, and have taken a step in this direction by presenting a synchronous grammar-based semantic parser. Drawing this connection facilitates a greater flow of ideas in the research community, allowing semantic parsing to leverage ideas from other work with tree automata, while making clearer how seemingly isolated efforts might relate to one another. We lose nothing in terms of performance by relying on these general formalisms, with results that are competitive with or better than the state of the art on a standard data set, but gain significantly in terms of modeling flexibility and ease of implementation. Once the parsing and training framework itself is implemented, a one-off investment, any number of models can be designed and tested without ever needing to invent a new algorithm. In fact, the model is closely related to the hybrid tree model of Lu et al. (2008), but where they found it necessary to develop several novel algorithms specifically tailored to their model we have relied on the general parsing and inference procedures outlined in Chapters 4 and 5 which are applicable to a large class of models. Situating the model in this general framework makes it easier to extend the work in this chapter on semantic parsing to our ultimate goal of implementing our new word learning model presented in the next chapter.

---

similar to that of UBL improved performance.

TEST	geo880 - 600 train/280 test							
	German				Greek			
	Acc		F1		Acc		F1	
UBL-S	**75.**	**0**	**75.**	**0**	73.	6	73.	7
WASP	65.	7	• 74.	9	70.	7	• **78.**	**6**
re-hybrid	62.	1	68.	5	69.	3	74.	6
scfgVB-hand	• 74.	6	74.	6	•**75.**	**4**	75.	4
	English				Thai			
UBL-S	**82.**	**1**	**82.**	**1**	66.	4	66.	4
WASP	71.	1	77.	7	71.	4	75.	0
re-hybrid	76.	8	• 81.	0	73.	6	76.	7
scfgVB-hand	• 79.	3	79.	3	• **78.**	**2**	• **78.**	**2**
	geo250 - 10 fold cross-val							
	English				Spanish			
UBL-S	80.	4	80.	6	79.	7	80.	1
WASP	70.	0	80.	8	72.	4	81.	0
re-hybrid	74.	8	82.	6	78.	8	• **86.**	**2**
scfgVB-hand	• **83.**	**2**	• **83.**	**2**	• **80.**	**0**	80.	0
	Japanese				Turkish			
UBL-S	**80.**	**5**	80.	6	74.	2	74.	9
WASP	74.	4	• **82.**	**9**	62.	4	75.	9
re-hybrid	76.	8	82.	4	66.	8	• **77.**	**5**
scfgVB-hand	• 78.	0	78.	0	• **75.**	**6**	75.	6

Table 6.4: Accuracy and F1 score comparisons on the geo880 and geo250 test sets. Highest scores are in bold, while the highest among the *tree based* models are marked with a bullet. The dotted line separates the tree based from non-tree based models.[3]

Chapter 7

Frog Stories Corpus: Language and Context

Marchman and Slobin collected data from over 100 subjects, children and adults, in several different languages to assemble what is sometimes referred to as the Frog Stories Corpus (Berman and Slobin, 1994). The corpus consists of transcribed narratives, describing the events visually depicted in the wordless picture book "Frog, Where Are You?" by childrens book author Mayer (1969). The original objective of the data collection was to study the development of narrative as children matured, but it captured the imaginations of numerous psychologists, resulting in the addition of narratives in several more languages, and was used to study a wide range of developmental phenomena, partially collected in two volumes (Berman and Slobin, 1994; Stromqvist and Verhoven, 2004).

The book tells the story entirely through a sequence of 24 black and white line drawings (see Figure 7.1 for a sample). Because the book is wordless, it permits the telling of the story with endless variations, and each narrator tells a slightly different version of events. At the same time, the pictures tell a clear enough story that a fair amount of consistency is maintained across speakers and even languages, making for an appealing data set for machine learning experiments as well as the already existent psychological studies. In particular, we are interested in modeling child language learners, and focus on the adult narratives to simulate child-directed speech, of which there are 12 for each language. We also focus on three of the languages, English, German, and Turkish, as these are the three with the largest amount of data, about 1,000 utterances per language. The corpus was originally gathered for the purpose of studying narrative development, where narrative complexity was measured by the

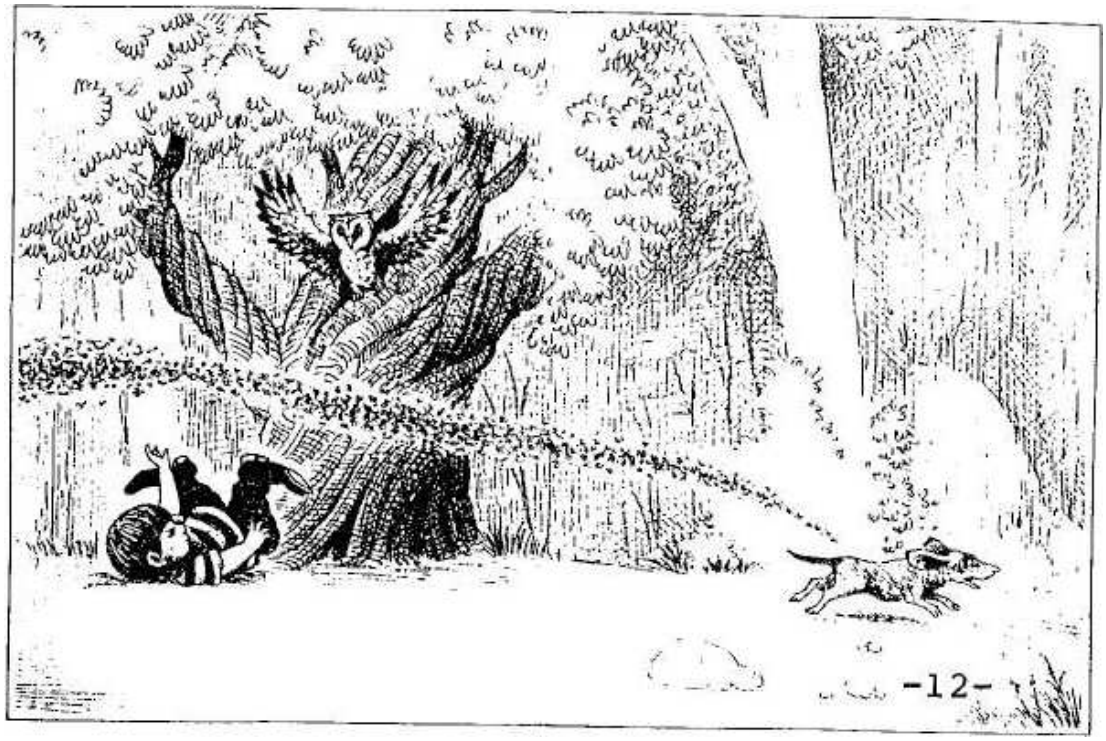

Figure 7.1: A sample picture from Mercer Mayer's wordless picture book "Frog, Where Are You?" (Mayer, 1969).

total number of events related over the course of the story. Thus, the original coding of the data was fairly event-centric, where utterances were transcribed and manually segmented so that each utterance contains at most one event. The original encoding of the data, segmented event by event, made it easy to count events, but otherwise there was little semantic analysis of the utterances.

The objective of our new semantic corpus annotations is to provide a data set for testing simulations of child learners, where the child is assumed to learn word meanings from the pairing of non-linguistic scenes (pictures in the book) with the words of the language. For this purpose, we have added two levels of annotation: a logical description of each of the 24 scenes in the book and meaning representations for each utterance of the three languages. We also provide English translation for the German and Turkish utterances, performed by PhD student volunteers at Macquarie University. The intention is that, using our annotations, a model can simulate a child's experience of trying to learn new words by listening to the narration while simultaneously observing the pictures of the book, and meaning inferences can be tested by comparing

Utterances			
	English	German	Turkish
words	6.8	6.5	4.2
events	0.9	0.9	0.9
entities	1.3	1.1	1.1
mods	0.1	0.1	0.1
roles	1.7	1.5	1.2
vertices	2.6	2.4	2.1
edges	1.6	1.4	1.2
utts/scene	38.6	52.9	35.4
total utts	927	1270	850

Table 7.1: Frog Stories corpus: utterances per language.

Scenes	
events	46.3
entities	26.6
mods	7.9
roles	114.4
vertices	74.0
edges	108.0
re-entrancies	14.8
components	7.3
num dags	11
total	24

Table 7.2: Frog Stories corpus: scenes.

them to the gold annotations. The utterance- and scene-level annotations both consist of a basic Neo-Davidsonion style semantics represented using the language of predicate calculus expressions, describing the action and the roles of the various actors and subjects of the action, a “who did what to whom” style representation. The scene descriptions and meaning representations follow the same conventions, and use the same symbolic language, a deliberate choice intended to facilitate the modeling of logical relationships between scene and individual utterances.

The annotation work for all three languages was performed by the author, working initially from English translations in the case of German and Turkish. Thus, the annotations are likely somewhat biased toward the English translation, although we made some effort to correct for this in subsequent passes over the data, using a bilingual dictionary to check for consistency with the vocabulary of the original language.

The resultant corpus with its annotations is similar in some respects to other corpora that have been used for semi-supervised semantic parsing such as the work of Chen and Mooney (2008) or Kwiatkowski et al. (2012). Chen and Mooney (2008), for instance, use machine-extracted summaries of a soccer game simulation for the non-linguistic context where each sentence in a sportscast narrative is assumed to correspond to an individual event. In contrast to the Frog Stories, their Robocup Sportscast corpus uses meaning representations that are generated independently from the natural

Concept/Word Types

		words		
	concepts	en	de	tr
events	315	278	437	408
entities	148	121	193	304
mods	64	14	30	24
roles	23	-	-	-
other	-	374	558	530

Table 7.3: Frog Stories corpus: number of concept types and the word types the word types that correspond to them for each language.

language narrative, which could potentially complicate the learning problem beyond what is permitted in the Frog Stories due to the fact that our meaning representations and scene descriptions are both based on the natural language narrative, potentially enforcing a tighter match between language and meaning than one might otherwise expect. However, the complexity of the natural language in Robocup Sportscast is very simple and repetitive with little variability and a small vocabulary of about 300 word types in a corpus of approximately 2000 sentences. Furthermore, the language of meaning representations is finite, permitting only a few hundred different semantic expressions which are themselves very simple and relatively flat. There are only 9 different possible event types where about 70% are of the form *pass(src-player, dst-player)* and only 30 possible entities. Finally, the scenes in Robocup Sportscast are also small with very limited ambiguity so that on average there are only about two meaning candidates per sentence.

Kwiatkowski et al. (2012), on the other hand, use a corpus based on the CHILDES Eve corpus (MacWhinney, 2015) where meaning representations are automatically derived from syntactic parses. This choice allows them to cheaply derive semantic annotations for a considerably larger data set of about 14 thousand naturalistic utterances with much more variability and complexity than Robocup Sportscast. However, the corpus lacks annotations specifically relating to the non-linguistic context at the time of utterance. Thus, the authors are forced to approximate scene descriptions by arbitrarily defining a set of meaning candidates for each utterances consisting of the meaning rep-

resentation for the utterance and its immediately preceding and following utterances. The corpus is also monolingual (Sportscast contains English and Korean narrations), making it impossible to test how well models might generalize across languages.

Frog Stories bears similarities to both corpora but with some unique characteristics largely centered on the scene descriptions. Like syntax-derived meaning approximations of Kwiatkowski et al. (2012), our meaning representations are also somewhat language dependent, where annotations for German and Turkish are loosely based on English translations. Still, despite the English bias, it is possible to test a model on different languages, unlike with the Eve corpus. In contrast, this language language dependence is somewhat less apparent in Robocup Sportscast where sentence "meanings" are actually derived automatically from non-linguistic events in a sports simulator. Also like Kwiatkowski et al. (2012), and in contrast to Chen and Mooney (2008), meaning representations of Frog Stories bear a resemblance to syntactic dependency analyses. However, Frog Stories meaning representations are more abstract, shedding much of the detail a syntactic parse would contain, thus forcing a learner to work slightly harder to map directly between words and meanings.

The biggest departure from both the Robocup Sportscast and CHILDES Eve corpora, however, is in that scene descriptions, which, rather than simple sets of meaning candidates, are structured representations of a large range of things that may be said about a picture in the children's book. One way these descriptions differ from the sets of meaning candidates in the case of Kwiatkowski et al. (2012) is candidates consist only of things that are actually said, whereas the Frog Stories scenes permit many utterances that, while logically consistent with the things that *were* said may not, in fact, correspond to any utterance itself.

Using a corpus based on a picture book also allows for an intuitive break-point to indicate where non-linguistic context begins and ends for each utterance. In the case of both Sportscast and Eve, experiments are forced to rely on arbitrary windows to identify meaning candidate sets, where the size is chosen in both cases based on the computational and learning performance of the model being tested. However, Frog Stories relies on a model-independent factor to define a scene, where a scene is constructed to cover all utterances associated with a single page of the picture book.

Finally, these scene descriptions also contain discourse-level information, identifying coreference across utterances, and capture alternative descriptions of the same entities, where a single entity may be described as a gopher, a squirrel, a mole, or simply by pronoun "he". Although we make limited use of this coreference information,

it could prove of interest for future work in modeling discourse level contributions to the word learning problem.

7.1 Truth-conditional semantics

In the original annotations, each utterance was identified with the picture it describes. As part of this dissertation work, we have added an extra level of annotation, coding the utterance meanings in the tradition of Neo-Davidsonian style truth conditional semantics (Lepore and Ludwig, 2007), with predicate logic expressions as meaning representations such as the following:

(7.1) the little boy fell from the tree

$$\exists e, x_1, x_2.\text{fall}(e) \wedge \text{patient}(e, x_1) \wedge \text{boy}(x_1) \wedge \text{source}(e, x_2) \wedge \text{tree}(x_2) \wedge \text{little}(x_1).$$

In keeping with the Davidsonian tradition, we say a logical expression represents the meaning of a natural language utterance if it evaluates to true exactly when the utterance itself is true and not otherwise. By convention, events, entities, and other qualifying features such as *little* are represented by unary relations and binary relations identify relationships between concepts such as the thematic relations identifying the agent or patient of a particular action.

The compositional nature of such expressions permits us to describe a large number of utterances with a relatively small set of relations. Neo-Davidsonian semantics tends to be event-centric, where unary relations like *fall* act as the main idea or pivot linking entities via their thematic relations. Thus, Neo-Davidsonian semantics seems like a particularly appropriate choice for representing utterances in the event-centric frog stories corpus. Neo-Davidsonion semantics often leads to semantic analyses that are similar to the syntax, where events are usually realized as verbs, entities as nouns, and the thematic relations as arcs in a dependency analysis, facilitating reasoning about the syntax/semantic interface, another topic central to our focus.

Because we are primarily interested in modeling child language acquisition where the representations learned are likely to be fairly simple, incomplete, and somewhat crude, we dispense with many of the finer nuances of the semantic theory, focusing on the high level "who did what to whom" notion of meaning that one might expect a preverbal child to infer from listening to a narrative while observing an interaction among non-linguistic entities. We largely omit things like tense, aspect, and mode and entirely ignore issues of quantification and scope. All variables are existentially quantified, and

we will often omit the quantifier itself in our examples since it is implicit. Also, proper names receive no special treatment; while the boy may be referred to as “Alex” in the narrative, we simply treat this name as an alternative word for *boy*.

7.1.1 Thematic relations

The binary thematic relations specify the role of each entity in the event. For instance, in example 7.1, *patient* and *source* specify that *boy* is the one falling and he is falling from *tree*. The thematic relations form a closed set consisting of 21 different types such as *agent*, *patient*, *experiencer*, *theme*, *loc*, *source*, *goal*, *recipient*, *time*, and *instrument*.

Also, sometimes entities are related to each other without the direct intervention of a mediating event or action. For instance, one may describe the location of one entity in relation to another.

the frog is inside the jar
$\text{loc-in}(x_1, x_2) \wedge \text{frog}(x_1) \wedge \text{jar}(x_2)$.

To handle this case of location, we appropriate a thematic relation *loc*, normally used to describe the location of an event and generalize it to describe locations for entities as well.

Possessives are another variety of inter-entity relation which we represent by a simple binary relation *pos*.

the elk’s antlers
$\text{pos}(x_1, x_2) \wedge \text{elk}(x_1) \wedge \text{antlers}(x_2)$.

In some cases, these thematic relations are also annotated with sub-types, such as in our example of “the frog is inside the jar”, where *-in* has been appended to *loc* to specify a particular type of location relation, distinct from *loc-behind* in the following example:

the frog behind the log
$\text{loc-behind}(x_1, x_2) \wedge \text{frog}(x_1) \wedge \text{log}(x_2)$

7.1.2 Pronouns

We annotate pronouns according to the information that the particular word encodes. For instance, pronouns in English may encode gender, number, and animacy.

he popped out of the hole
$\text{pop-out}(e) \wedge \text{agent}(e, x_1) \wedge \text{a-m-sg}(x_1) \wedge \text{source}(e, x_2) \wedge \text{hole}(x_2)$

the child climbed onto it
$\text{climb}(e) \wedge \text{agent}(e, x_1) \wedge \text{child}(x_1) \wedge \text{dest-on}(e, x_2) \wedge \text{sg}(x_2)$

The use of the word "he" indicates that the subject is male, singular, and animate, a combination of features which we encode by the constant *a-m-sg*, while "it", on the other hand, is neuter, singular, and possibly inanimate, getting the constant *na-sg* specifying that it is singular with indeterminate animacy. The exact information encoded in a pronoun is language dependent. For example, Turkish, in contrast to English, lacks a direct counterpart for "he" (*a-m-sg*), ""she" (*a-f-sg*), or "it" (*na-sg*), and instead has a pronoun "o" which can be used for animate, inanimate, male, or female entities, and only specifies number (singular), which we denote by *sg*.

To keep things simple, we omit possessive pronouns such as "his" or "my" from the annotations. Also, in the case of Turkish, a pro-drop language, we asked the translator to note where pronouns were supplied in translation that were not present in the original transcription and omitted these from the semantic annotations as well.

7.1.3 The lexicon

The particular set of unary relations and constants chosen to annotate a given utterance is chosen based on the content words of the utterance. Specifically, in addition to the utterance level annotations, the corpus also includes a lexicon which identifies the words each constant and unary relation corresponds to in the corpus. The lexicon is carefully constructed so as to enforce a soft mutual-exclusivity principle in keeping with the literature on word learning, so that for each language every event concept corresponds to a single verb stem, and similarly for entities and so on.

7.2 Scene descriptions

Since we are interested in modeling the learning of language meanings from scenes, we also annotate the pictures of the picture book. These are also rendered as expressions in predicate logic, much like the Neo-Davidsonian logical forms of the natural language, a kind of world semantics. In fact, in an effort to remain consistent with the utterance-level annotations, we rely on essentially the same conventions we did there, using

Neo-Davidsonian semantics to describe the images as well as the individual utterances that describe them.

There is much we are abstracting away from in our choice of scene annotation scheme. Much as a learner must work his way backward through the lexical and syntactic levels from surface form to meaning representation, there are similar layers of inference and interpretation between the raw sensual input of a picture and what a learner eventually perceives as the scene. For instance, there are questions of saliency, which features or objects stand out over others and are worthy of mention. Just identifying a collection of raw features as a coherent object is no trivial task. There are also principles of social perception that govern how a line drawing might resolve to an animate character, and by which motivations and intentions are assigned to this character. Children must solve all of these problems as well.

Most of these interesting perceptual problems are beyond the scope of our work here, however. Instead, we pick up where the perceptual machinery has largely left off, assuming that the learner has already rendered the picture into some representation conducive to natural language description. This is an obvious oversimplification of the problem the child actually faces, but the enormous complexity of language acquisition plus perception insists on some form of simplifying assumptions. The many interesting challenges of artificial vision are far from solved, and in many respects fall short of the levels one might expect a human language learner to perform at, so, in some ways, an assumption of a greater level of competence than we can currently achieve with artificial vision technology has the potential to yield a more realistic model of human cognition.

Of course, our situated semantic parsing model cannot entirely ignore the mapping between world and utterance. However, we carefully circumscribe the relationship between scene and utterance by enforcing two assumptions. First, we assume that anything any narrator, across all languages, ever says is a true statement about the scene. Second, we make a closed world assumption, assuming that anything not mentioned by any of the narrators is untrue. Thus, we abstract away from almost all the perceptual problems with the exception of some aspects of salience: the learner must still resolve which aspects of the scene each utterance describes.

Given these assumptions of the relationship between scene description and utterance meaning, and the utterance-level annotations, it is possible with some additional effort to work backwards to derive the scene description. We first compile the list of utterances across all languages that describe a given scene. Then we combine the

meaning representations into a single logical description of the scene so that each scene description is guaranteed to subsume the meaning of each of its associated utterances. I.e., each scene description entails every utterance in the narration about the corresponding picture in the book. The guiding principle is to find the simplest subsuming expression that only entails true statements about the scene.

For example, given the meaning representations of four different utterances

der Junge ist vom Baum hinuntergefallen (German)
/ The boy has fallen down off the tree /
$\text{fall}(e) \wedge \text{patient}(e,x_1) \wedge \text{boy}(x_1) \wedge \text{source}(e,x_2) \wedge \text{tree}(x_2)$

the boy fell to the ground (English)
$\text{fall}(e) \wedge \text{patient}(e,x_1) \wedge \text{boy}(x_1) \wedge \text{dest}(e,x_3) \wedge \text{earth}(x_3)$

the boy disturbed the owl in his tree (English)
$\text{disturb}(e) \wedge \text{agent}(e,x_1) \wedge \text{boy}(x_1) \wedge \text{patient}(e,x_4) \wedge \text{owl}(x_4)$
$\wedge \text{loc}(e,x_2) \wedge \text{tree}(x_2)$

arilar kucuk kopegi kovaliyor (Turkish)
/ the bees are chasing the little dog /
$\text{chase}(e) \wedge \text{agent}(e,x_5) \wedge \text{bees}(x_5) \wedge \text{theme}(e,x_6) \wedge \text{dog}(x_6) \wedge \text{little}(x_6)$

the scene description will contain something like the following

$$\begin{aligned}&\text{fall}(e_0) \wedge \text{patient}(e_0,x_1) \wedge \text{boy}(x_1) \wedge \text{source}(e_0,x_2) \wedge \text{dest}(e_0,x_3) \wedge \text{earth}(x_3) \\ &\wedge \text{disturb}(e_1) \wedge \text{agent}(e_1,x_1) \wedge \text{patient}(e_1,x_4) \wedge \text{owl}(x_4) \wedge \text{loc}(e_1,x_2) \wedge \text{tree}(x_2) \\ &\wedge \text{chase}(e_2) \wedge \text{agent}(e_2,x_5) \wedge \text{bees}(x_5) \wedge \text{theme}(e_2,x_6) \wedge \text{dog}(x_6) \wedge \text{little}(x_6)\end{aligned} \tag{7.2}$$

Constructing the scene requires resolving a coreference problem where we identify the variable modified by *boy* in the first utterance with that of the *boy* in the second and third, and similarly we note that the *fall* event is the same in both the first and second utterances.

7.2.1 Entity coreference

Often coreference resolution is trivial given the semantic representation, since the constant *boy* always refers to the same character throughout the story. However, in many cases, the same entity may be described very differently. Figure 7.2 illustrates an example where the same entity is conceptualized as both a *hamster* and a *gopher*, and is

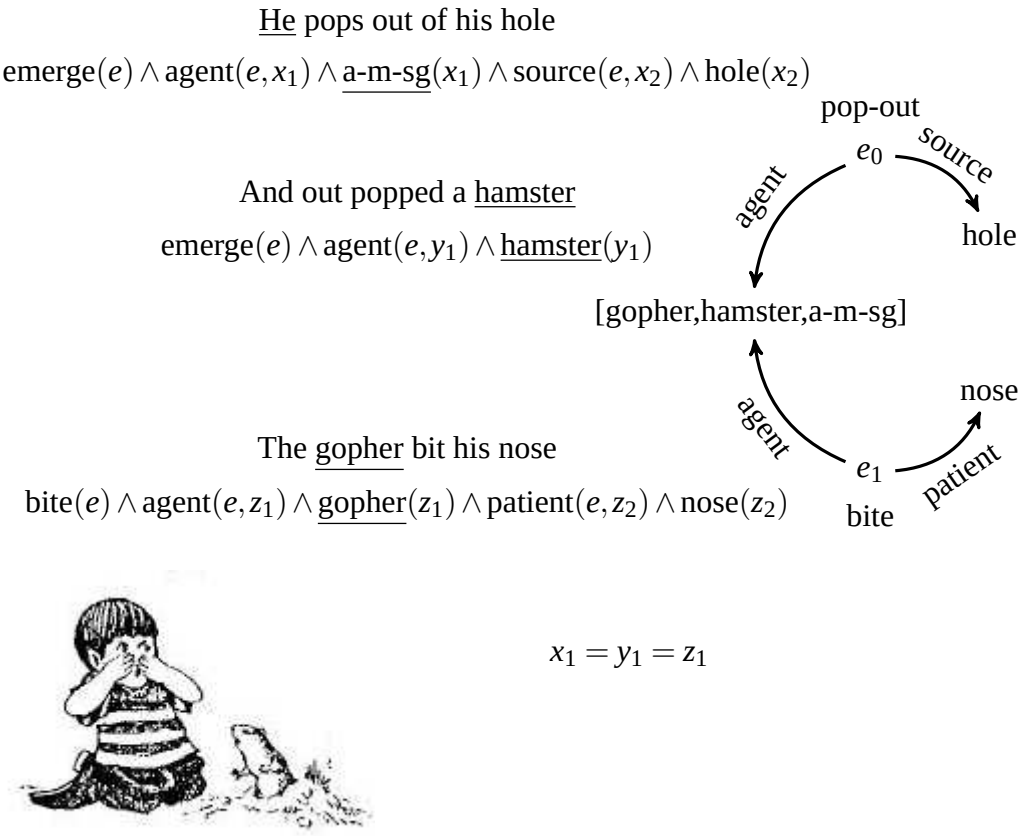

Figure 7.2: Linking variables that correspond to the same entity.

sometimes referred to by pronoun. Since hamsters and gophers are semantically different, even if they all refer to the same animal in the story, we annotate each differently, but add notes to indicate that they share the same referent. This same strategy generalizes to tracking the referents of pronouns. Thus, each scene has a set of equivalence classes indicating which concepts and pronouns refer to the same entities in the story. For the example of the figure, the variable the *gopher*, *hamster*, and *a-m-sg* pronoun all refer to the same entity, resulting in the equality:

$$x_1 = y_1 = z_1.$$

We indicate the equality by enforcing that the same variable name is used across all utterances for the scene. That is, we essentially raise the utterance-specific existential quantifiers to the scene level to cover the full set at once.

Similarly, sometimes different referents will be described the same way. In one scene there are multiple frogs, a father and mother frog, each of which my be referred

to as *frog*, even though they are different entities. In this case, we distinguish the different frogs by using different variables just as we do for any other dissimilar concepts. Furthermore, they are described as “he” and “she” depending on which one is being talked about. So for the male frog, we have

$$\text{frog}(x_0) \text{ and a-m-sg}(x_0)$$

and for the female frog we have

$$\text{frog}(x_1) \text{ and a-f-sg}(x_1)$$

where $x_0 \neq x_1$.

7.2.2 Event coreference

In constructing the example scene in the opening example of this section 7.2 we had to solve a similar coreference problem for the event *fall*

> the boy fell from the tree
> $\text{fall}(e) \wedge \text{patient}(e,x_1) \wedge \text{boy}(x_1) \wedge \text{source}(e,x_2) \wedge \text{tree}(x_2)$
>
> the boy fell to the ground
> $\text{fall}(e) \wedge \text{patient}(e,x_1) \wedge \text{boy}(x_1) \wedge \text{dest}(e,x_2) \wedge \text{earth}(x_2)$

to arrive at

$$\text{fall}(e) \wedge \text{patient}(e,x_1) \wedge \text{boy}(x_1) \wedge \text{source}(e,x_2) \wedge \text{tree}(x_2) \wedge \text{dest}(e,x_3) \wedge \text{earth}(x_3).$$

We treat the two utterances as partial descriptions of the same event and thereby construct the full description by computing their conjunction.

Additionally, just as different entity referents may share the same semantic type, the same goes for events. Consider the following example:

> the boy looks at the frog
> $\text{look}(e) \wedge \text{experiencer}(e,x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e,x_2) \wedge \text{frog}(x_2)$
>
> the frog looks at the dog
> $\text{look}(e) \wedge \text{experiencer}(e,x_1) \wedge \text{frog}(x_1) \wedge \text{theme}(e,x_2) \wedge \text{dog}(x_2)$

If these were two (incomplete) instances of the same event, the scene would contain this single logical expression

$$\begin{aligned}&\text{look}(e) \wedge \text{experiencer}(e,x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e,x_2) \wedge \text{frog}(x_2)\\ &\quad \wedge \text{experiencer}(e,x_2) \wedge \text{theme}(e,x_3) \wedge \text{dog}(x_3)\end{aligned}$$

implying that the following would also be true

* the frog looks at the frog
$\text{look}(e) \wedge \text{experiencer}(e,x_2) \wedge \text{frog}(x_2) \wedge \text{theme}(e,x_2)$

* the boy looks at the dog
$\text{look}(e) \wedge \text{experiencer}(e,x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e,x_3) \wedge \text{dog}(x_3)$

However, neither of these are part of a valid description of the story. To prevent such over-generalizations when we form the scene description, we use different event variables to distinguish distinct events of the same type:

the boy looks at the frog
$\text{look}(e_0) \wedge \text{experiencer}(e_0,x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e_0,x_2) \wedge \text{frog}(x_2)$

the frog looks at the dog
$\text{look}(e_1) \wedge \text{experiencer}(e_1,x_2) \wedge \text{frog}(x_2) \wedge \text{theme}(e_1,x_3) \wedge \text{dog}(x_3)$

where $e_0 \neq e_1$.

Thus, the scene would be

$$\begin{aligned}&\text{look}(e_0) \wedge \text{experiencer}(e_0,x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e_0,x_2) \wedge \text{frog}(x_2)\\ &\quad \wedge \text{look}(e_1) \wedge \text{experiencer}(e_1,x_2) \wedge \text{theme}(e_1,x_3) \wedge \text{dog}(x_3)\end{aligned}$$

which entails the true statements but not the false ones.

Given these referent equivalence classes for the entities and events, we can easily assemble the individual utterances into scene descriptions. Assembly merely involves eliminating duplicate relations involving the same variables.

7.3 Graphs

There is a long tradition of representing logical expressions in graphical terms, and it is still in active development. See Figure 7.3 for some examples of just a few different approaches. The conceptual graphs of Sowa (1976), for instance, represent the

logical predicates as nodes in a graph, and draw edges between nodes corresponding to the sharing of variable arguments. Entity-relation diagrams for describing the tables of relational databases are very similar to the conceptual graph convention. The dependency-based compositional semantics of Liang et al. (2011), though limited to trees, follows a similar scheme of representing predicates as nodes and arguments as edges. It is also possible to represent predicates by edges, such as in the semantic dependency graphs of Titov et al. (2009); Martin and White (2011). Semantic dependency graphs are designed to represent semantic dependencies between words, much as the edges in syntactic dependency graphs. In yet another example, the discourse representation structure graphs of Le and Zuidema (2012) represent both variables and predicates by nodes, drawing edges from variable nodes to the predicates that take them as arguments.

We choose a graphical representational scheme that closely relates to the semantic dependency graphs (see Figure 7.3(c)), where

- variables are identified with vertices,
- binary relations specify edges, directed from the vertex identified with the left argument to that of the right argument,
- vertex and edge labels are identified by their associated relations, where unary relations specify vertex labels and binary relations specify edge labels.

Thus, the graph represented in Figure 7.3(c) can be translated into the following expression:

$$\begin{aligned} &\text{climb}(e_0) \wedge \text{agent}(e_0, x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e_0, x_2) \wedge \text{tree}(x_2) \\ \wedge &\text{disturb}(e_1) \wedge \text{agent}(e_1, x_1) \wedge \text{patient}(e_1, x_3) \wedge \text{owl}(x_3) \wedge \text{loc}(e_1, x_2) \\ \wedge &\text{chase}(e_2) \wedge \text{agent}(e_2, x_4) \wedge \text{bees}(x_4) \wedge \text{theme}(e_2, x_5) \wedge \text{dog}(x_5) \wedge \text{little}(x_5). \end{aligned}$$

Our approach differs from the semantic dependency graphs of Titov et al. (2009); Martin and White (2011) principally in that we do not identify nodes directly with words, since we are interested in a more general scheme that would allow us to graphically represent scene descriptions where word nodes do not make sense. Also, a major part of the phenomena we wish to model is the learning of the mapping between predicates and words and we do not want to start from the assumption of an identity relation given directly in the meaning representation. Note that semantic and syntactic dependency graphs, though not identical, often closely resemble one another by virtue of

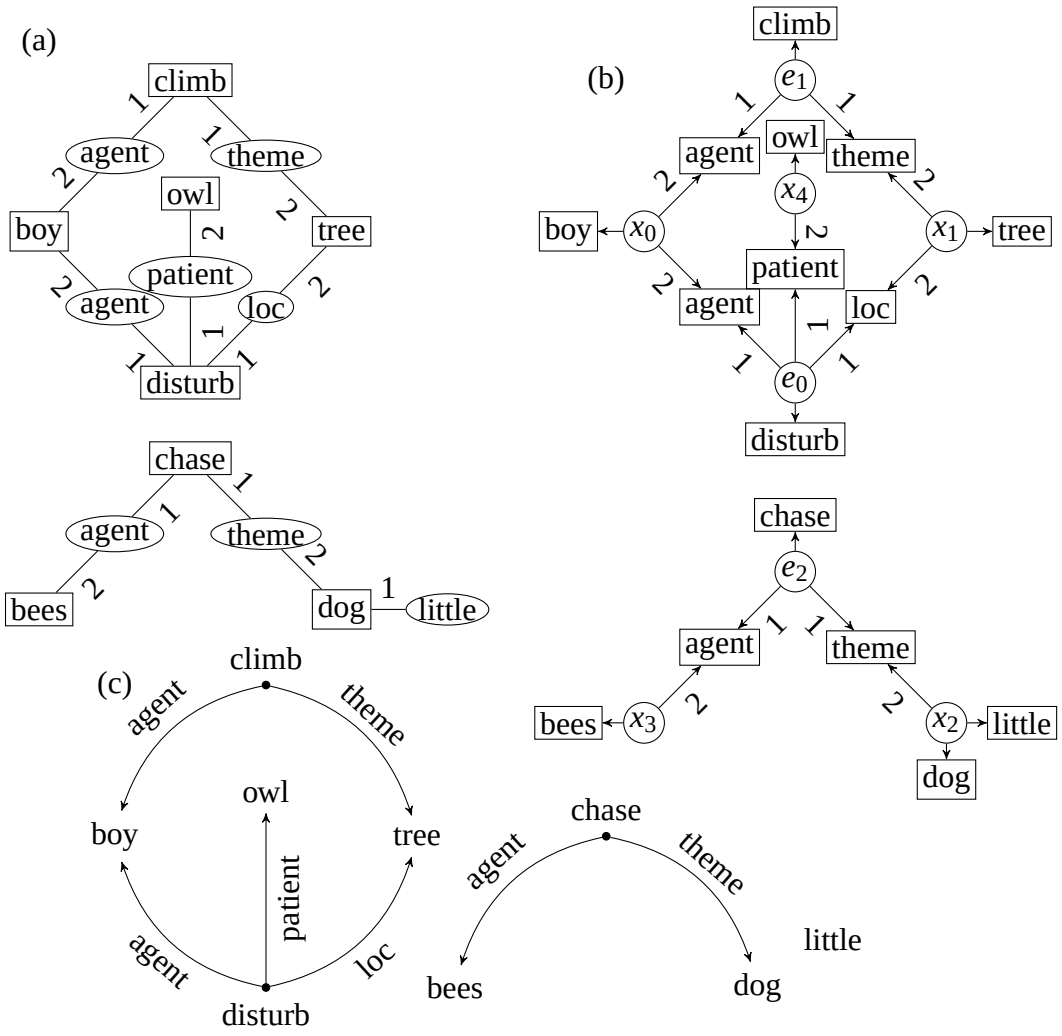

Figure 7.3: Three different graph representations of the expression in equation 7.2. (a) A conceptual graph. (b) A discourse representation structure graph (global wrapper node omitted for clarity). (c) Our representation scheme.

the close relationship between thematic relations and syntactic arguments. In fact, semantic dependency graphs often feature in work at the syntax-semantics interface. It is for this close correspondence between syntax and semantics, as well as the relative simplicity of the representation, that we choose a similar approach for our work here.

Coreference can be represented in the graph by identifying each referent with a single vertex of the graph. If there are semantically different ways of referring to the referent, these become multi-labeled vertices, one label for each descriptive type. For instance, in Figure 7.2, the rodent that emerges from the ground to bite the boy on the nose is called a “hamster,” a “gopher,” and sometimes is just referred to as “he,”

resulting in three different labels for the corresponding vertex.

7.4 Encoding scenes as forests

A graph is capable of representing an arbitrary expression in our subset of predicate calculus, but this expressivity comes at computational cost. Compared to trees and tree languages, the set of tools for processing more general classes of graph is considerably more limited, and those that exist tend to be expensive, less well understood, and difficult to implement. It is easier to implement efficient algorithms for parsing trees, on the other hand, and a general graph parser seems somewhat overpowered considering that the vast majority of utterances in our corpus can be described by trees. In fact, there are only 12 out of a total of 3030 utterances whose meaning cannot be represented by trees. Of course, even if meanings can usually be represented by trees, scene descriptions are invariably more complex graphs, but these graphs can be approximated using multiple trees aggregated into a forest. We can then parse these forests using the highly optimized tree parser presented in Chapter 4. Inevitably the resultant forests are larger than the more compact graph representations (by about 30%), leaving it unclear exactly how much forestization saves over parsing the graph directly using a similarly optimized graph parser. However, we can still justify forestization over direct graph parsing considering that the optimizations described in Chapter 4, while still theoretically applicable, are considerably more difficult to implement efficiently in the general graph parsing setting.

To “forestize” the graphs, there are two features that we must enforce. The first is rootedness, i.e., every node of each subtree must be reachable by following a directed path from some root node. The second is the single parent property where every node except the root which has only outgoing edges must have exactly one incoming edge (i.e., no reentrancies). We enforce these two properties in the Frog Stories scene graphs using a semi-automated process with some manual intervention for handling reentrancies.

Rather than construct the scene graphs and then forestize these graphs, the process starts by first enforcing the tree property at the individual meaning representations, then joining these meaning representation trees into subgraphs at using the coreference annotations. Finally, we make another pass over these subgraphs to again enforce the tree properties, forming the forest.

We start by enforcing the rootedness and single parent properties in each meaning

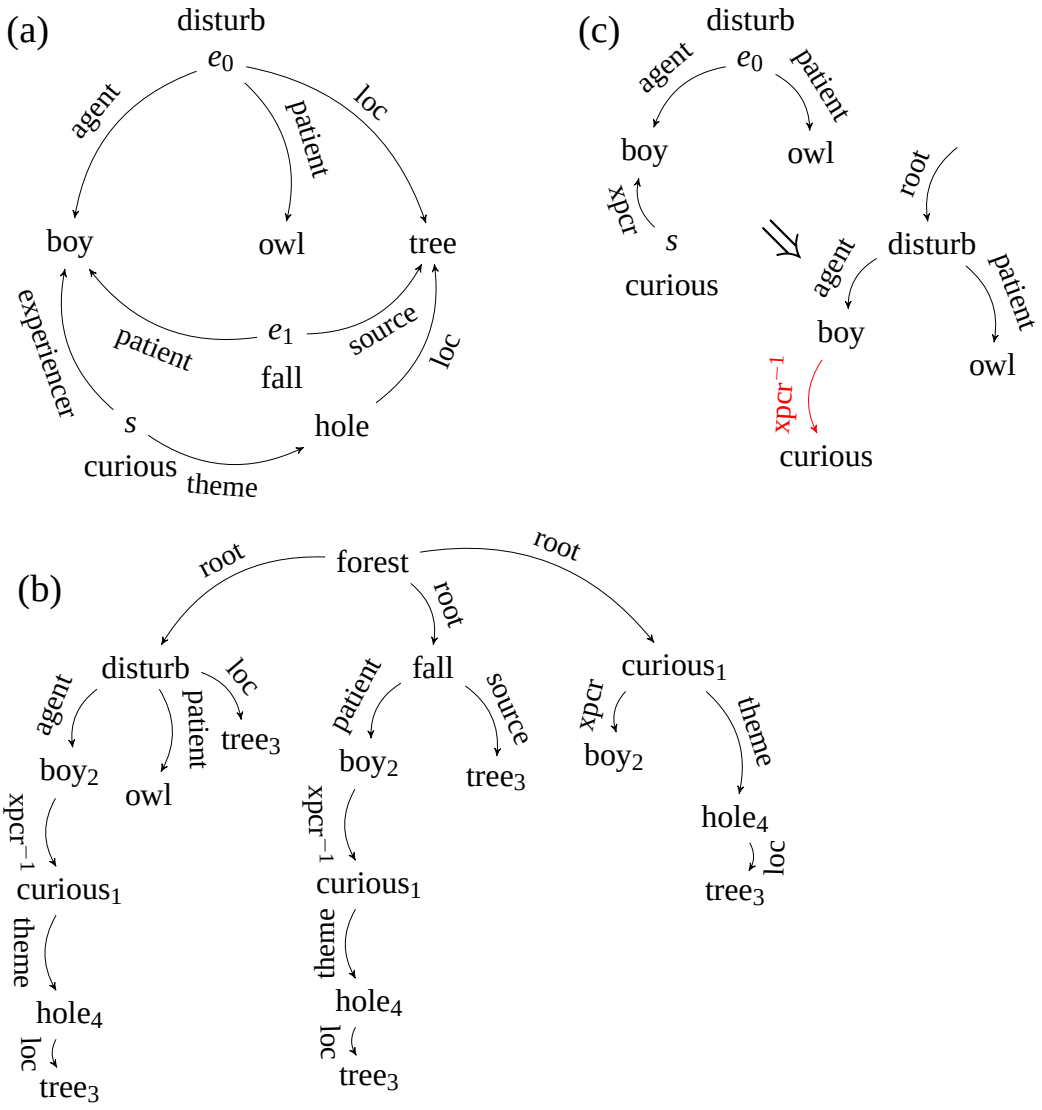

Figure 7.4: (a) A scene graph. (b) A forest approximation, one event per tree. Indices indicate duplicate nodes required for preserving the tree property. (c) The transformation of a meaning representation into a rooted tree.

representation. First, we pick the node to make root by identifying by prioritizing events first, and then affects like *happy* or *angry* over entities. There is typically only one event per utterance, given the way the original corpus was encoded, but in the case of affects or entities, occasionally there may be multiple candidates for the root. In such cases, we score each candidate by calculating its out-degree minus its in-degree and choose the one with the highest score, breaking ties arbitrarily. Once the root is identified, we invert all of its incoming edges and then invert the minimum number of additional edges in the graph necessary so that all other vertices are reachable from the

root by some directed path. To preserve the semantics of these reversed edges, we add an extra notation in the edge label so that, for instance, reversing edge experiencer(e,x) results in experiencer$^{-1}(x,e)$. Figure 7.4(c) illustrates an example where *disturb*, as the event node, is chosen as root and the *experiencer* edge is reversed so that it has a directed path to *curious*.

Next, we remove the reentrancies from the meaning representations, of which there are only 12. We do so by splitting the vertex by introducing one or more additional vertices, one per in-edge, and dividing up the incoming edges among them, leaving the out-edges attached to the original vertex.

Once we have guaranteed that the meaning representations of the individual utterance is a tree, we proceed to construct the maximal trees of the scene forest. These trees are essentially the maximal frames of the scene, constructed by taking the union of the roles of each instance of the root concept (event, affect, or entity) in the utterances. The union operation may introduce new reentrancies which can again be removed by splitting vertices and distributing the incoming edges among them until there are no more reentrancies. Outgoing edges may also need to be duplicated depending on whether every meaning representation is covered by the graph, which we enforce manually and test with a script. Occasionally cycles may be introduced during the union operation, which we also break by splitting vertices and redistributing edges to guarantee that every utterance meaning is included as a subtree, another step which enforced manually and tested with a script.

These trees then make up the forest, which itself can be represented as a single tree by adding an extra root node. Figure 7.4(b) illustrates the resultant forest representation corresponding to the scene graph fragment in part (a). Note that *curious* along with the accompanying *theme* and its *loc* are duplicated as well as appearing as root of its own tree. This allows for utterances such as

> "the boy is curious about the hole"
> $\text{curious}(s) \wedge \text{experiencer}(s,x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(s,x_2) \wedge \text{hole}(x_2)$

as well as utterances corresponding to the tree in Figure 7.4(c) such as

> "the curious boy disturbed the owl"
> $\text{disturb}(e) \wedge \text{agent}(e,x_1) \wedge \text{boy}(x_1) \wedge \text{patient}(e,x_2) \wedge \text{owl}(x_2)$
> $\wedge\ \text{curious}(s) \wedge \text{experiencer}(s,x_1)$

The resultant forest is potentially much larger than the graph would be, with computational implications since the parsing algorithm presented in Chapter 4 takes time

directly proportional to the size of the tree. However, in practice we find that the necessary vertex splitting and edge duplication leads to forests containing about 1.29 times as many vertices and 1.1 as many edges as the unforestized graphs. General graph parsing typically requires far more than time quadratic in the graph being parsed, usually exponential in the degree or tree width of the graph such as the algorithm of Chiang et al. (2013), so our forest transformation potentially saves us some computational expense, at least in terms of the asymptotic bounds. However, it is difficult to say exactly how much the forestization step actually saves in practice without a direct run-time comparison, and a fair comparison would require first implementing the optimizations described in Chapter 4 in a general graph parser, something we leave for future work.

7.5 Quantifying and constraining ambiguity

A scene description is a compact way of defining a set of possible true statements about the scene, and we can always expand the representation to an explicit set by simply enumerating all possible logically entailed statements. Unrestricted, however, this set would be astronomically large, essentially the powerset of all concepts in the scene with roughly an average of 2^{200} entailments per scene, and very few of these entailments actually correspond to anything someone is likely to actually say. For instance, it is hard to conceive of a statement with the following meaning representation

$$\text{agent}(e_0, x_1) \wedge \text{boy}(x_1) \wedge \text{curious}(s) \wedge \text{source}(e_1, x_2) \wedge \text{tree}(x_2).$$

The expression is merely a jumble of disconnected concepts; the boy is an agent of some event, the tree the source of some other, and it is unclear what *curious* has to do with either. Even the most efficient computational model would also likely need to employ some sort of filter just to keep the problem tractable, and it seems likely that human learners employ biases or constraints that allow them to quickly dismiss such nonsensical meaning candidates. Thus, a word learning system would likely do well to constrain this set of entailments to something that is more computationally tractable, focusing attention on only the more plausible possibilities.

We employ five main constraints.

- **Single Event:** Utterances can contain at most one event. This assumption is not only guaranteed by the fact that utterances were manually segmented by the original encoders so there would be at most a single event per utterance (Berman

and Slobin, 1994), it might also be a reasonable assumption in general since utterances in child-directed speech are often relatively short, perhaps consisting of a single clause. Such a simplifying assumption could greatly reduce processing overhead both for human and computational learners, particularly in the early stages of word learning when there are few known words to rely on disambiguating novel words.

- **Connectedness:** Everything in the meaning representation either plays a direct role in the main event of the utterance or is related to something that is through a chain of binary relations. Representing the scene as a graph or tree, this means that all meanings must be connected subgraphs or subtrees. The connectedness restriction reflects an intuition that entities that interact in the scene are more likely to also be talked about in the same utterance than are entities that seem to have no relationship. This constraint is related to the conditions for well-formedness in Lexical Functional Grammar (LFG), where *coherence* dictates that every grammatical function must be licensed by some predicate in the sentence (Bresnan, 2001).

- **Relation Completeness:** If an entity or event is omitted, all binary relations that include it, either as a left or right argument, must also be omitted. For instance, excluding *boy* from the meaning representation corresponding to the scene in Figure 7.4(a) would also exclude the incident *agent*, *patient*, and *experiencer* edges. While connectedness relates to the coherence criterion for well-formedness in LFG, this constraint that relations must be fully specified relates to *completeness*, which states that a sentence containing a particular predicate must also contain all of its required grammatical functions (Bresnan, 2001).

- **Rooted Tree:** Meanings must be tree shaped (i.e., no concept can play more than one role). Additionally, this tree must be rooted such that there is some node (i.e., concept) from which all others can be reached by following a directed path. Again, this assumption matches the data since there are only a dozen utterances out of roughly 3000 that contain reentrancies. Also, assuming the scene description has been encoded as a forest as described in Section 7.4, it is easy to enforce the rootedness constraint. This constraint has a relatively small impact on the total number of meaning representations, but can aid in computational modeling since, as mentioned in Section 7.4, there are more algorithmic tools

	unpruned	pruned
max	1,333	352
avg	339	183
min	82	33

Table 7.4: The number of meanings entailed by the scene description with and without frequency pruning. In the frequency-pruned scenes all but the most frequent 20 entities and 20 events have been removed from the scene and meaning representations.

(which are also usually more easily implemented) for efficient processing of tree structures than for most other varieties of graph.

- **Singly Labeled:** Every vertex of the tree has exactly one label. In our graphical language, adjectives and adverbs like *little* and *quickly* are realized as additional labels on the entity or event predicate vertex. However, for our work we are primarily interested in noun learning, and multi-labeled trees present additional complexity, so we further simplify the problem by removing such modifiers from the scene. There may still be multi-labeled vertices in the *scene* description due to varying ways of referring to the same entity (see the gopher example in Figure 7.2), but only one of these may be chosen for a specific meaning representation.

It seems reasonable that human word learners might employ, if not these exact constraints, at least some similar sort of simplifying assumptions. The first three in particular are fairly plausible as constraints human learners might place on the problem at early stages of learning. Furthermore, most of the constraints are justified by the data since they turn out to be true for all but a few utterances in the Frog Stories corpus.

Employing these constraints, we arrive at much more manageable numbers. The first column of Table 7.4 lists the maximum, minimum, and average number of meanings for the 24 scenes of the Frog Stories corpus. In one scene, there are well over a thousand meaning candidates even under this fairly constrained setting, a number that even the dynamic programming-based word learner in Chapter 8 is unable to handle.

Psychologists often appeal to the notion of saliency to explain the ability of children to quickly home in a manageable set of candidates. Certain things are more likely

to be discussed than others, either because they stand out perceptually and draw the storyteller's eye or figure more prominently in the story. For instance, a child might exploit an understanding of the typical structure of story narrative and pay greater attention to the actions surrounding the main characters than to more peripheral action like a bird flying in the distance. Focusing on the more central aspects of the story could allow a learner to grasp the basic narrative even with an imperfect understanding of the language used. Such a focus could aid in the learning of words for the more central concepts even as the less central ones are ignored, thereby simplifying the learning problem to something more manageable.

One would expect this kind of scene-level saliency to correlate with the actual descriptions produced by the storytellers themselves, so we can approximate it using frequency counts where the more salient actors and actions are mentioned more frequently. In line with this idea, Table 7.4 also reports the ambiguity numbers where all but the most frequent event and entity types are pruned from the scene and meaning representations. Relying on saliency in this way, and pruning pronouns and a few of the more abstract concepts, we restrict the corpus to just 20 different event types and entity types, reducing the number of meaning candidates in our most difficult scene from over a thousand down to about 350, and down to 183 candidates on average. Even in this further constrained set, however, there are on average well over four times as many meaning candidates as actual utterances per scene from which a child or model must learn, suggesting that the learning problem is still likely to prove quite challenging.

In fact, in spite of the simplifying assumptions and saliency-based pruning, there is still far more ambiguity in the training data than typically assumed in either psychological or computational experiments where the learner usually only chooses from a handful of concepts in the non-linguistic context. That is, while relying on many of the same ideas, the annotations of our corpus are far less constraining than typically assumed. Chapter 8 explores the effect of this added ambiguity in testing whether a statistical learner is capable of learning under these conditions.

Chapter 8

Word Learning

Infants are confronted with a challenging problem when it comes to learning the meanings of the words they hear in the arbitrary stream of sounds spoken around them, but they nevertheless seem to be quite effective at leveraging the many sources of information available to work the problem out. In particular, the value of co-occurrence patterns between words and objects across varying non-linguistic contexts has been demonstrated repeatedly both through computational modeling and behavioral experiments, a phenomenon known as cross situational learning. Typically, the non-linguistic context in both the computational and behavioral experiments consists of a set of isolated objects that varies across utterances. By assuming that words refer to objects in the immediate environment, the learner can use the fact that a particular word is more likely to be heard in the presence of a particular object to deduce that the word refers to the object. There are many computational models that operate on this principle (Frank et al., 2009; Yu and Ballard, 2007; Fazly et al., 2010; Jones et al., 2010). Of course, the world is much richer than portrayed in such simulations where the learner essentially only has one kind of information, the variation in the presence or absence of words and objects across utterances and scenes. This chapter presents a cross situational learning model that exploits an additional source of information: relational information among the words in the utterance and among the non-linguistic concepts in the scene. Concept-concept relations are represented explicitly in the scene description, while word-word relations are latent, left to be inferred by the model.

There are different ways that such relational information could influence the learning problem. For one, relations between entities in the scene may directly shape the learner's hypotheses about the meanings of descriptive utterances. For instance, one might assume that things that interact in the scene are more likely to be referred to in

the same sentence than two completely unrelated entities. Thus, focusing on connected sub-components of the scene could have a pruning effect on the space of subsets, helping to guide learning and simplify search. For instance, if the scene portrays a swarm of bees chasing a dog with an owl and trees in the background, it seems plausible that bees and dog might appear within the same utterance, while the trees and owl with which the dog and bees do not directly interact are less likely to be included. A learner has much less to judge from in the case where the scene is represented as a completely unstructured set of concepts any of which may or may not be related and, thus, any subset is as plausible as any other. If one must consider the full powerset of concepts, inference can quickly become intractable, and any plausible pruning strategy based on their relationships could prove invaluable in terms of tractability as well as the quality of learning outcomes.

Secondly, structured representations of the non-linguistic context also provide a more subtle source of cross situational information, since in a relational setting, situations vary not only by the presence or absence of entities but also by the particular relations among them. Consider the example sentence in Figure 8.1. Mere co-occurrence statistics for the concepts *boy* and *dog* and the words in the following two sentences

(8.1) $\text{lick}(e) \wedge \text{agent}(e, x_1) \wedge \text{dog}(x_1) \wedge \text{theme}(e, x_2) \wedge \text{boy}(x_2)$
the dog licked the boy

(8.2) $\text{scold}(e) \wedge \text{agent}(e, x_1) \wedge \text{boy}(x_1) \wedge \text{theme}(e, x_2) \wedge \text{dog}(x_2)$
the boy scolded the dog

provides no information since both are present in each situation. Co-occurrence statistics might help infer the meanings of the verbs “licked” and “scolded”, but there are other useful patterns such as the fact that the agent occurs at the beginning and the theme at the end that a learner could exploit.In fact, this relational information about agents and themes and subjects and objects in English could assist in the learning of the nouns even if the learner failed to identify the verb meanings.

This correspondence between relations and word order can be modeled by exploiting the similarity between word learning and semantic parsing to integrate the key features of both into a single model. Semantic parsers seek to learn compositional meanings for whole sentences rather than isolated words and concepts, and already make heavy usage of such relational information. Word learners, on the other hand, often deal with a much greater degree of referential ambiguity, since semantic parsers

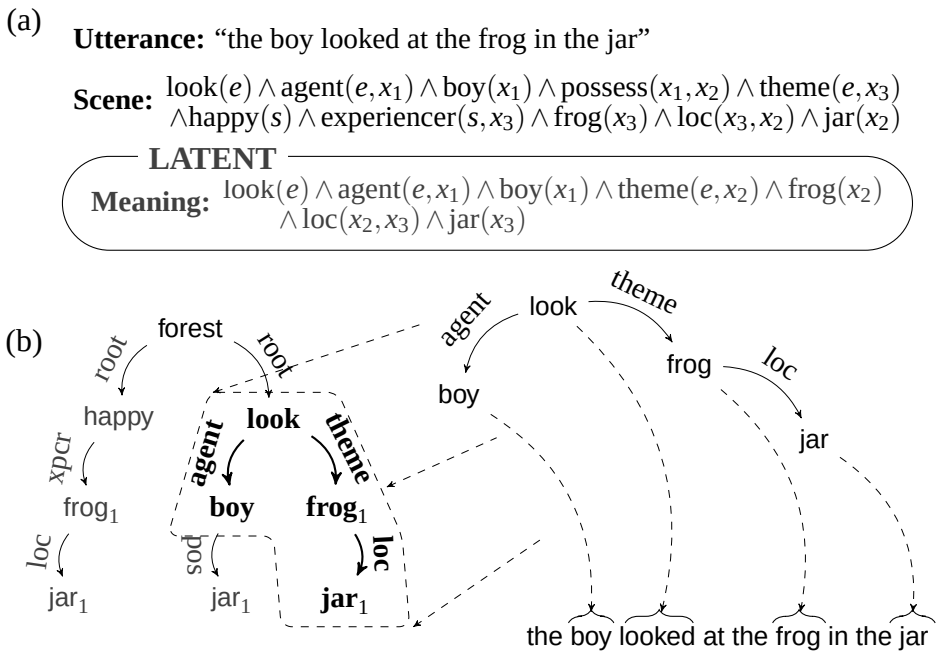

Figure 8.1: (a) An example utterance, its latent meaning representation, and the accompanying scene description and (b) a mapping from words to scene via the meaning.

are trained on observed meaning-sentence pairs. Word learners must instead simultaneously infer from context alone the meaning of the sentence as well as the meanings of individual words.

Others have proposed similar solutions by using semantic parsers as word learners while replacing the single observed gold meaning by a set of possible candidate meanings (Börschinger et al., 2011; Kwiatkowski et al., 2012; Chen and Mooney, 2008; Kim and Mooney, 2010). However, this approach has been limited by issues of computational complexity, forcing models to work with relatively small sets of half a dozen or so candidate meanings. We propose to expand this set of meaning candidates by representing it as a language in its own right, defined by a probabilistic grammar, which allows us to apply dynamic programming techniques to effectively search and perform inference over the set in a far more efficient manner. The grammar-based framework presented in Chapters 3 through 5 makes it relatively straightforward to adapt and incorporate the semantic parsing model of Chapter 6, and is also well-suited to implementing such a language-centric approach to representing sets of meaning candidates.

The key idea is to represent the non-linguistic context (or *scene*) by a single log-

ical structure as described in Chapter 7 rather than a collection of unrelated meaning representations and then search over the possible meaning candidates for statements about the scene. Figure 8.1(a) presents an example of the learning scenario based on the Frog Stories corpus described in Chapter 7: the learner hears the utterance and observes a scene and must infer the utterance's meaning during the process of learning the contributions of the individual words to that meaning. Part (b) of the figure illustrates the relationship among the words, meaning, and scene, where the meaning serves as a bridge explaining which aspects of the scene the words are describing.

As with the simpler semantic parsing model of Chapter 6, the word learner is implemented as a single synchronous grammar that jointly derives scene, meaning, and words. The meaning-to-words relationship is modeled by a variant of the semantic parsing grammar, but we add a third dimension to the grammar so that the yield of each derivation is a triple: scene representation, meaning representation, and words. Because the entire model is implemented as a grammar it is relatively easy to incorporate syntactic aspects for joint inference with word learning to explore the impact of syntactic bootstrapping. In fact, the semantic parsing model already incorporates this kind of joint syntactic-semantic learning. Coupled with an ability to manage a larger number of meaning candidates, our fully grammar-based approach permits us to better explore the strength of the influence of various effects such as joint syntactic learning on resolving referential ambiguity during word learning.

With greater ambiguity, we can measure this strength more effectively than previously possible, asking the question, "from where do the learning biases and constraints come?" Are they a necessity of the learning problem, i.e., information theoretic limitations, or are they constraints necessitated mainly by the cognitive resources of the learner? If a computational model can learn under less constrained settings, one leans toward ruling out the first possibility, leaving cognitive constraints as the more likely explanation.

However, while the grammar framework allows a model to explore more ambiguous learning scenarios, there are fundamental challenges that we can only hope to ease, not entirely avoid. As psychologists have argued, it is likely that human learners rely on cognitive biases and exploit a combination of alternative sources of information to further simplify the task. We do not go so far as most previous work, however, in assuming that these biases narrow the space of possible hypotheses to a dozen (or often fewer) candidate meanings per utterance, since this would defeat the purpose of the exercise. Imposing such constraints or biases, however, does not not necessar-

ily eliminate that much ambiguity. In fact, the corpus analysis of Chapter 7 suggests there may still be hundreds of candidates remaining even after heavily constraining the problem. Instead, we explore a marriage of the two, constraining biases plus greater computational power, to address a greater degree of referential ambiguity.

The model incorporates both soft statistical properties to guide learning and hard constraints to make the task computationally tractable. Like most statistical models of word learning, we rely on cross-situational consistency to home in on reasonable word-concept pairs. We implement it as a Bayesian model, using a sparse prior on word-concept pairs in the lexicon encouraging the learner to adopt a kind of mutual-exclusivity-like bias for small lexicons. Furthermore, without feeding the model any additional language-specific information, we allow the model to learn and exploit simple syntactic cues based on canonical word orderings found in the training data. These soft constraints provide the statistical learner with the ability to gradually home in on a lexicon, but there are additional hard computational limitations that such soft constraints do little to mitigate. In particular, in our grammar framework more ambiguity translates directly into terms of larger packed parse forests and heavier memory costs, and a highly ambiguous scenario can quickly exhaust memory resources, necessitating the introduction of several additional hard constraints.

We enforce four main types of hard constraints. Section 7.5 already outlined the structural constraints, some of which are closely related to the connectedness and coherence properties of LFG (Bresnan, 2001). Additionally, although we relax the assumption of previous work that salience and shared attention eliminates most of the ambiguity, we still find a place for them here. Psychologists have observed that speakers often employ intonation to highlight content words, helping to focus children's attention and distinguish them from other background words (Fernal and Mazzie, 1991). Thus, we distinguish content words in the input to our system as well. Furthermore, shared attention may serve to direct the learner's attention to the most salient events and actors in a story, which we implement through scene pruning as described in Section 7.5. This scene pruning can also be seen as a kind of theory of mind (Tomasello, 2001), where the learner follows the story and comes to anticipate which entities and events are most likely to be mentioned. Finally, since we are primarily interested in interactions with early stage syntactic learning, which is likely to be more easily measured after the learner has acquired at least a small set of words, we also simulate this prior knowledge by providing the model with a small seed lexicon. Of course, even after incorporating these additional constraints, there is still considerable ambiguity

remaining, with hundreds of meaning candidates per utterance, more than sufficient to explore our main questions of whether the stricter constraints assumed in previous work are necessary for learning to occur at all or are simply a product of cognitive limitations.

8.1 Model Definition

Word learning in the Frog Stories scenario consists of the learner listening to a story told by a narrator while examining and collating this description with the story told through the images in the picture book and other non-linguistic cues. The modeling objective is to explore how a learner might infer word meanings by reasoning about the parallel sources of input to determine which words refer to which aspects of the scene. This reasoning process might manifest as a sequence of guesses: the learner listens to a sequence of utterances, makes a guess as to the meaning of each as it comes, checks this guess against the scene to test whether it makes sense in the current context, and then performs some analysis akin to a semantic parser to resolve which words relate to which components of the proposed meaning. We model this procedure probabilistically as a generative process where the model first guesses a meaning, generates a mapping from meaning to scene (i.e., verifies that the guess is consistent with the context), and then proceeds to generate the words conditioned on the meaning. The process can be conceptualized as a two stage procedure for first generating a scene-to-meaning map and then the meaning-to-words map (although the particular order of the stages can be reversed without changing anything).

The meaning-to-scene map is governed by several simplifying assumptions. First, the model assumes that every utterance is true, and second, that the scene description contains everything that one may wish to describe, i.e., a closed world assumption. Thus, in order to verify the validity of a guess for the utterance meaning, the model merely needs to check whether the scene description contains at least one instance of the meaning representation expression as a subset. While this simplifies the task considerably, it still leaves a number of possible meanings exponential in the size of the scene description, so we further restrict the model to only propose meanings that are connected subtrees of the scene description forest. Because of the way these forests are constructed (see Chapter 7), this has the further effect of enforcing an assumption that every meaning candidate includes at most one event. Given these assumptions, the map from meaning to scene consists of deterministically generating a copy of the meaning

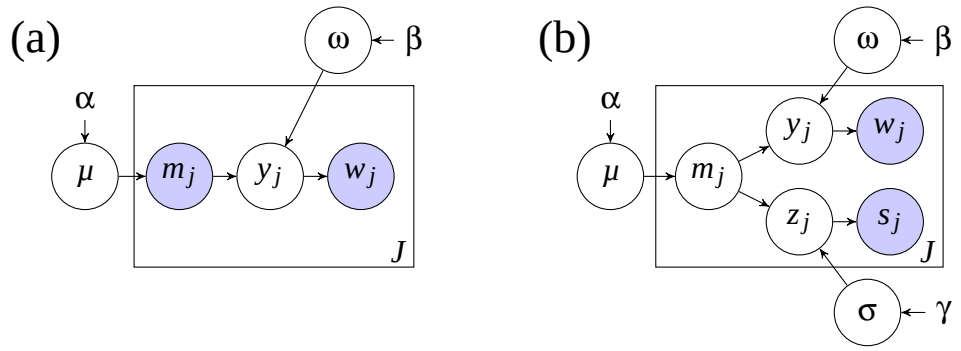

Figure 8.2: The generative model for (a) the semantic parser described in Chapter 6 and (b) the word learner for a corpus of J utterances. w_j is an utterance, m_j is its corresponding meaning representation, and s_j is the scene that the utterance is describing. Variables y_j and z_j map from meaning to words and meaning to scene, respectively, as governed by the probabilistic synchronous grammar. μ is the set of multinomial parameters for the language model over meaning representations m, ω is the set of multinomial parameters for the utterances w given their corresponding meaning representation, and pi are the parameters for multinomial parameters for the meaning-to-scene mapping. α, β, and γ are the parameters for their respective Dirichlet priors.

representation and then randomly adding onto this seed to generate the remainder of the scene.

The map from meaning to words is very similar to the semantic parsing model of Chapter 6, and consists of choosing words with which each component of the meaning is expressed and determining a linear order in which to arrange these words.

Figure 8.2(b) presents a plate diagram describing the model. The meaning representation m is drawn from a product of multinomials with parameters μ which are drawn from Dirichlet priors with parameters α. This meaning representation is then checked against the observed scene by generating a map as per another product of multinomials with parameters σ drawn from another set of Dirichlet distributions with parameters γ. Finally, the words are generated in more or less the same fashion as done for the semantic parser. In fact, part (a) presents the semantic parsing model for comparison, demonstrating that, at this high level, it is essentially a subset of the model

where m is observed during training. Equation 8.3 summarizes the model.

$$P(m,w,y,s,z,\mu,\omega,\sigma|\alpha,\beta,\gamma) = P(\mu|\alpha)P(\omega|\beta)P(\sigma|\gamma) \cdot \prod_{j=1}^{J} P(m_j|\mu)P(w_j,y_j|m_j,\omega)P(s_j,z_j|m_j,\sigma) \tag{8.3}$$

- $P(m_j|\mu)$ is the probability over plausible meaning candidates given a basic knowledge about what sorts of things are likely to appear in a given relation to one another. For instance, frogs are more like to hop than to bark, a fact that can be learned by the model and encoded in the parameters μ.

- $P(w_j,y_j|m_j,\omega)$ describes the conditional probability of the map to words given a particular meaning representation, dictating which words and word orders are most likely when expressing a particular concept or combination of concepts.

- $P(s_j,z_j|m_j,\sigma)$ describes the conditional probability of the scene given that same meaning representation, modeling our closed world assumption and our assumption that every utterance is a true statement. This distribution also encodes inferred information about which aspects of a scene are least likely to be talked about and instead be generated as background information.

Since the model is implemented as a multi-weighted synchronous grammar, these three distributions can be explicitly defined in terms of the weights on the rules in a derivation. For consistency with the semantic parsing chapter, assume $\mu(r)$ and $\omega(r)$ are the products of the weights of rule r pertaining to the meaning and the words, respectively. Additionally, let $\sigma(r)$ be the product of weights relating to the scene. Then we can compute all three by simply multiplying the weights of the rules in a particular derivation x_j.

$$P(m_j|\mu) = \prod_{r \in x_j} \mu(r)$$

$$P(w_j,y_j|m_j,\omega) = \prod_{r \in x_j} \omega(r)$$

$$P(s_j,z_j|m_j,\sigma) = \prod_{r \in x_j} \sigma(r)$$

The precise definition of these probability distributions will be covered in the following sections, but like in the case of the semantic parser in Chapter 6, each rule may contribute to all three portions of the yield simultaneously and therefore have a weight

vector that includes a μ, ω, and σ, which can be implemented as a multi-weighted grammar as defined in Section 3.4.

However, before diving into the implementation details we would like to first acknowledge that there are, of course, other ways of formulating the word learning problem, and some that may even seem more intuitive to some readers than the one presented here. We could, for instance, first generate the scene and only then generate the meaning representation while conditioning on the scene instead of the guess-and-check approach we have taken where first the meaning is independently generated and then the scene is generated conditioned on the meaning. When considering alternative approaches, however, it is important to keep in mind that our ultimate objective is to learn a correspondence between the meaning and words, i.e., the joint distribution of m, y and w. The probability over the scene s and the related multinomial variables and parameters are only necessary for training, and are not used directly in computing the lexicon itself. The particular factorization chosen in Equation 8.3 makes it straightforward to compute this joint probability over m, y and w while integrating out the additional variables pertaining to the scene. In fact, because of the conditional independence between the words and the scene, one can simply drop s, z, and pi, resulting in the basic semantic parsing model shown in Figure 8.2(a). In an alternative model where the meaning is generated conditioned on the scene, we might, for example, have the following factorization:

$$P(w,m,s) = P(s)P(m|s)P(w|m).$$

In this case, information about the meaning m is conflated with information about the scene s so that one would need to somehow sum over all possible scenes (of which there are an infinite number under our scene-generating grammar).

$$P(w,m) = \int P(s)P(m|s)P(w|m)ds = P(m)P(w|m)$$

This is not necessarily an insurmountable obstacle since, assuming the model were implemented as a grammar, one could likely utilize a variant of the inside-outside algorithm to perform this integration. Our formulation, however, allows us to forgo this extra complexity altogether. Furthermore, our model design is simplified by the close relationship to the semantic parsing model which would otherwise require refactoring to accommodate the alternative scene-first approach.

We now turn to the implementation of the model in terms of grammar rules and weights.

$$\text{N[ROOT]} \rightarrow \lambda e.\text{look}(e) \wedge \text{E[LOOK/}\underline{\text{AGENT}}\text{]}(e) \wedge \text{E[LOOK/}\underline{\text{THEME}}\text{]}(e) \quad \text{(m1)}$$

$$\text{N[LOOK/}\underline{\text{AGENT}}\text{]} \rightarrow \lambda x.\text{boy}(x) \quad \text{(m2)}$$

$$\text{N[LOOK/}\underline{\text{THEME}}\text{]} \rightarrow \lambda x.\text{frog}(x) \wedge \text{E[FROG/}\underline{\text{LOC}}\text{]}(x) \quad \text{(m3)}$$

$$\text{N[FROG/}\underline{\text{LOC}}\text{]} \rightarrow \lambda x.\text{jar}(x) \quad \text{(m4)}$$

$$\text{E[LOOK/}\underline{\text{AGENT}}\text{]} \rightarrow \lambda e.\text{agent}(e,x) \wedge \text{N[LOOK/}\underline{\text{AGENT}}\text{]}(x) \quad \text{(m11a)}$$

$$\text{E[LOOK/}\underline{\text{THEME}}\text{]} \rightarrow \lambda e.\text{theme}(e,x) \wedge \text{N[LOOK/}\underline{\text{THEME}}\text{]}(x) \quad \text{(m11b)}$$

$$\text{E[FROG/}\underline{\text{LOC}}\text{]} \rightarrow \lambda x_0.\text{loc}(x_0,x_1) \wedge \text{N[FROG/}\underline{\text{LOC}}\text{]}(x_1) \quad \text{(m12)}$$

Table 8.1: A grammar that generates the meaning in Figure 8.1. Node-generating rules (top) yield event and entity predicates while edge-generating rules (bottom) yield the role labels.

8.1.1 Meaning generation: $P(m|\mu)$

The generative process starts off by generating the meaning representation, much as the semantic parser in Chapter 6 does. First, we choose a label for the root node and then proceed to generate the thematic role labels for its child edges conditioned on the root label. These thematic roles are filled with entities or other events of their own, each generated independently conditioned on its parent and role type, and generation continues in recursive fashion, choosing each node label conditioned on its parent and role type and then generating its own children in turn by first selecting their number and type conditioned on the node label. In this way we descend, generating the tree until terminating at the leaves.

Table 8.1 shows the grammar fragment for generating the meaning representation in example 8.1. Nonterminals of the form N[...] expand to generate event and entity concept labels on the nodes of the tree, while those of the form E[...] expand to produce the role types, represented as binary edges. Thus, the generation of the meaning representation of our running example begins with rule m1 generating a *look* event with an agent and theme. Rule m11a then yields the agent role label and then rule m2 produces *boy*. The *frog* is generated as a theme of *look* in much the same way by rules m11b and m3, and then the frog's location is specified by rules m12 and m4.

Multiplying the weights $\mu(r)$ for each step of the derivation defines the following probability distribution over meaning representations, where we assume e (for *entity* or *event*) stands for a node in the meaning representation:

$$P(m|\mu) = \prod_{e \in m} P(e|\textit{parent-role}(e), \textit{parent}(e), \mu) P(\textit{child-roles}(e)|e, \mu)$$

where

$$\begin{aligned} \textit{child-roles}(e) &= \text{the set of child roles of } e, \\ \textit{parent-role}(e) &= \text{the parent role of } e, \text{ and} \\ \textit{parent}(e) &= \text{the parent node predicate of } e. \end{aligned}$$

Rules such as m1 through m4 implement the main behavior of the probabilistic model where each rule is weighted in order to (1) generate the event or entity label and (2) the number and type of roles:

$$\begin{aligned} \mu(m1) &= P(\text{look}|\text{ROOT}, \mu) P(\text{agent}, \text{theme}|\text{look}, \mu) \\ \mu(m2) &= P(\text{boy}|\text{look}, \text{agent}, \mu) P(\emptyset|\text{boy}, \mu) \\ \mu(m3) &= P(\text{frog}|\text{look}, \text{theme}, \mu) P(\text{loc}|\text{frog}, \mu) \\ \mu(m4) &= P(\text{jar}|\text{look}, \text{loc}, \mu) P(\emptyset|\text{jar}, \mu) \end{aligned}$$

These rules have factors of the form

$$\varphi_\mu(r) = \langle \textit{lhs}(r), \textit{event}(r) \rangle \cdot \langle \textit{event}(r), \textit{child-roles}(r) \rangle,$$

where $\textit{lhs}(r)$ identifies the left-hand-side symbol, $\textit{event}(r)$ identifies the event unary on the right-hand side, and $\textit{child} - \textit{roles}(r)$ lists the roles as encoded in the nonterminals on the right-hand side. Rules m11a through m12 then produce the binary relation edge labels in deterministic fashion (after already being chosen probabilistically in the previous steps), with a factorization function of $\varphi_\mu(r) = \langle \emptyset, \emptyset \rangle$ leading to a weight of 1.

The process is very similar to that of the semantic parser described in Chapter 6, but involves a few additional independence assumptions to accommodate added variation in the Frog Stories corpus. For one, instead of conditioning each entity/event on the full signature of its parent (i.e., the parent node label and its full set of child role labels), each concept is only conditioned on its own role type and parent, irrespective of the number and roles of its siblings. Additionally, each node label and its child edge labels are chosen in two separate steps rather than all at once. These independence

assumptions help accommodate the greater degree of variability in the Frog Stories narrations over the considerably more formulaic sentences in GeoQuery. For instance, there can sometimes be multiple agents, or the agent may be omitted (especially in pro-drop languages like Turkish), while in GeoQuery a binary function will always have exactly two arguments.

8.1.2 Word generation: $P(w, y|m, \omega)$

Once the meaning is generated, the probabilistic model proceeds to generate the words conditioned on the meaning, walking from root to leaves and translating each concept in the meaning representation into a string of words. Again, the process is very similar to that of the semantic parser, but there are a few modifications to better match the objectives of the word learning task. In particular, our word learner only attempts to learn meanings for content words (roughly corresponding to what we will call *foreground* words), and the model maintains a separate *background* unigram distribution to account for function words. This distinction between foreground and background words helps the model learn a more compact lexicon than the semantic parsing model would which better matches the assumptions behind the construction of the gold lexicon, our target. While the introduction of the background unigram distribution is a departure from the semantic parsing model, translation from a concept in the meaning representation to a substring of the utterance proceeds in an otherwise very similar fashion. First the concept and its children are linearized into a particular word order pattern that determines where words of each type are to be inserted. The words themselves are then generated to fill in the details such that each concept is identified with exactly one word in the substring, drawn from a concept-specific distribution over foreground words, and the remaining words are selected from the universal background unigram distribution.

For example, the linearization of a node like *frog* in Figure 8.1 which has a single

child is drawn from the following eight possible patterns:

FG-WORD CHILD
BG-WORDS FG-WORD CHILD
FG-WORD CHILD BG-WORDS
BG-WORDS FG-WORD CHILD BG-WORDS
CHILD FG-WORD
BG-WORDS CHILD FG-WORD
CHILD FG-WORD BG-WORDS
BG-WORDS CHILD FG-WORD BG-WORDS

There are three sub-steps to choosing one of these particular linearizations, all conditionally independent. Specifically, if the local tree being linearized contains n children, these steps are

1. the ordering of children ($n!$ possibilities) conditioned only on the set of their role labels,

2. the placement of the single foreground word, which can be at the beginning or end or anywhere between the children with $n+1$ choices conditioned on the parent node label and the number of children, and

3. the placement of strings of background words which can occur anywhere between or at the beginning or end of the string of children with 2^{n+1} configurations, also conditioned on the parent node label and number of children.

The process is implemented by pairing the monolingual meaning representation grammar rules with word generating rules (shown in Table 8.2) and, while retaining the meaning generating weights $\mu(r)$, adding word-to-meaning translation weights $\omega(r)$ to each rule to jointly generate meaning and words. Rules w1a-w4b are examples of linearization rules, where rule w3b corresponds to the second pattern in the list above. The weights of this rule governing the meaning-to-word map $\omega(r)$ are defined such that

$$\omega(w3b) = P(\text{CHILD}|\{\text{loc}\},\omega)P(\text{FG-WORD }\_|\text{frog},1,\omega) \cdot P(\text{BG-WORDS }\_|\text{frog},1,\omega),$$

$$\text{N[ROOT]} \rightarrow \langle\ \lambda e.\text{look}(e) \wedge \text{E[LOOK/}\underline{\text{AGENT}}\text{]}_{\boxed{0}}(e) \wedge \text{E[LOOK/}\underline{\text{THEME}}\text{]}_{\boxed{1}}(e)\ \|\ \text{E[LOOK/}\underline{\text{AGENT}}\text{]}_{\boxed{0}}\ \text{W[LOOK]}\ \text{E[LOOK/}\underline{\text{THEME}}\text{]}_{\boxed{1}}\ \rangle \quad \text{(w1a)}$$

$$\text{N[LOOK/}\underline{\text{AGENT}}\text{]} \rightarrow \langle\ \lambda x.\text{boy}(x)\ \|\ \text{W[BG]}\ \text{W[BOY]}\ \rangle \quad \text{(w2b)}$$

$$\text{N[LOOK/}\underline{\text{THEME}}\text{]} \rightarrow \langle\ \lambda x.\text{frog}(x) \wedge \text{E[FROG/}\underline{\text{LOC}}\text{]}_{\boxed{0}}(x)\ \|\ \text{W[BG]}\ \text{W[FROG]}\ \text{E[FROG/}\underline{\text{LOC}}\text{]}_{\boxed{0}}\ \rangle \quad \text{(w3b)}$$

$$\text{N[FROG/}\underline{\text{LOC}}\text{]} \rightarrow \langle\ \lambda x.\text{jar}(x)\ \|\ \text{W[BG]}\ \text{W[JAR]}\ \rangle \quad \text{(w4b)}$$

$$\text{E[LOOK/}\underline{\text{AGENT}}\text{]} \rightarrow \langle\ \lambda e.\text{agent}(e,x) \wedge \text{N[LOOK/}\underline{\text{AGENT}}\text{]}_{\boxed{0}}(x)\ \|\ \text{N[LOOK/}\underline{\text{AGENT}}\text{]}_{\boxed{0}}\ \rangle \quad \text{(w11a)}$$

$$\text{E[LOOK/}\underline{\text{THEME}}\text{]} \rightarrow \langle\ \lambda e.\text{theme}(e,x) \wedge \text{E[LOOK/}\underline{\text{THEME}}\text{]}_{\boxed{0}}(x)\ \|\ \text{N[LOOK/}\underline{\text{THEME}}\text{]}_{\boxed{0}}\ \rangle \quad \text{(w11b)}$$

$$\text{E[FROG/}\underline{\text{LOC}}\text{]} \rightarrow \langle\ \lambda x_0.\text{loc}(x_0,x_1) \wedge \text{N[FROG/}\underline{\text{LOC}}\text{]}_{\boxed{0}}(x_1)\ \|\ \text{N[FROG/}\underline{\text{LOC}}\text{]}_{\boxed{0}}\ \rangle \quad \text{(w12)}$$

$$\text{W[BG]} \rightarrow \langle\ -\ \|\ \text{U[BG]}\ \text{W[BG]}\ \rangle \quad \text{(w20a)}$$

$$\text{W[BG]} \rightarrow \langle\ -\ \|\ \text{U[BG]}\ \rangle \quad \text{(w20b)}$$

$$\text{U[BG]} \rightarrow \langle\ -\ \|\ \text{the}\ \rangle \quad \text{(w30a)}$$

$$\text{U[BG]} \rightarrow \langle\ -\ \|\ \text{at}\ \rangle \quad \text{(w30b)}$$

$$\text{U[BG]} \rightarrow \langle\ -\ \|\ \text{in}\ \rangle \quad \text{(w30c)}$$

$$\text{W[LOOK]} \rightarrow \langle\ -\ \|\ \text{look}\ \rangle \quad \text{(w31)}$$

$$\text{W[BOY]} \rightarrow \langle\ -\ \|\ \text{boy}\ \rangle \quad \text{(w32)}$$

$$\text{W[FROG]} \rightarrow \langle\ -\ \|\ \text{frog}\ \rangle \quad \text{(w33)}$$

$$\text{W[JAR]} \rightarrow \langle\ -\ \|\ \text{jar}\ \rangle \quad \text{(w34)}$$

Table 8.2: Rules for generating the meaning-to-word map in Figure 8.1. The five categories of rule are for event type, role, background unigram stopping, and background and foreground word generation.

where FG-WORD _ and BG-WORDS _ denote the appearance of the foreground and background words at the start of the string (as opposed to _ FG-WORD which would denote the end of the string).

Each word is either drawn from one of various foreground distributions or from a single background distribution. Foreground words are generated one per node in the meaning representation by rules w31-w34, guaranteeing that every entity and event label in the meaning is represented in the utterance. In general, there is one rule for every concept-word pair, permitting the model to map each concept to any word in the utterance, where the gold alignment portrayed in Figure 8.1 is achieved by a derivation constructed from the rules in Table 8.2. These rules have weights such as $\omega(w31) = P(\text{look}|\text{look})$. Background words are generated in a similar fashion according to rules w20a-w30c, but where multiple words can be drawn to build up a multi-word string with stopping probability determined by the weights of w20a ($P(\textit{continue}|\omega)$) and w20b ($P(\textit{stop}|\omega)$) and w30a-w30b draw words from the distribution. Edge-generating rules such as w11a-w12 are deterministic and simply coordinate nonterminals in the meaning representation and word string.

The entire process is summarized by the following equation:

$$\begin{aligned} P(w,y|m,\omega) &= \prod_{e\in m} P(y_e = \langle \ell, \textit{fg-word}, \textit{bg-w}_1, ..., \textit{bg-w}_k\rangle | e, \textit{child-roles}(e), \omega) \\ &= P(\ell|e,\textit{child-roles}(e),\omega) P(\textit{fg-word}|e,\omega) \prod_{i=1}^{k} P(\textit{bg-w}_i|\omega) \end{aligned}$$

Variable y is a sequence of linearization and word generation steps required for translating meaning representation m into word string w, where y_e is the particular sequence required for translating e, made up of a single linearization step ℓ followed by the choice of foreground word *fg-word*, and zero or more background word substrings $\textit{bg-w}_1, ..., \textit{bg-w}_k$. Linearization ℓ is further broken down into the three steps of choosing the order of e's children ℓ^{args}, the position of the foreground word ℓ^{fg}, and the number and location of the background substrings ℓ^{bg}. If c is the number of e's children, the probability of ℓ factorizes as

$$P(\ell|e,\textit{child-roles}(e),\omega) = P(\ell^{args}|\textit{child-roles}(e),\omega) P(\ell^{fg}|e,c,\omega) P(\ell^{bg}|e,c,\omega)$$

where each of $P(\ell^{args}|\textit{child-roles}(e),\omega)$, $P(\ell^{fg}|e,c,\omega)$, and $P(\ell^{bg}|e,c,\omega)$ are multinomial distributions.

The foreground word is drawn from a multinomial distribution $P(\textit{fg-word}|e,\omega)$ specific to e's label, and the background substrings $\textit{bg-w}_1, ..., \textit{bg-w}_k$ are generated as

per the background unigram distribution, where each *bg-w*$_i$ is made up of n_i words *bg-word*$_{i,1}, \ldots,$ *bg-word*$_{i,n_i}$.

$$P(\textit{bg-w}_i|\omega) = \prod_{j=1}^{n_i-1} P(\textit{bg-word}_{i,j}|\omega)P(\textit{continue}|\omega) \cdot P(\textit{bg-word}_{i,n_i}|\omega)P(\textit{stop}|\omega).$$

To implement as a multi-weighted grammar, we define the following four feature functions:

- *event-roles*(r) identifying the event and its roles in the meaning portion of the right-hand side of rules such as w1a-w4b,
- $\ell(r)$ which identifies the linearization pattern in the word-generating portion of rules such as w1a-w4b,
- *stop*(r), a boolean function which returns true if and only if the rule is w20b, and
- *word*(r) which identifies the word on the right-hand side of rules such as w30a-w34.

With these feature functions, we can define the portion of the rule factorization function that deals with generating the words of utterances as follows:

$$\varphi_\omega(r) = \langle \textit{event-roles}(r), \ell(r)\rangle \quad \text{(w1a-w4b)}$$
$$\varphi_\omega(r) = \langle \emptyset, \emptyset\rangle \quad \text{(w11a-w12)}$$
$$\varphi_\omega(r) = \langle \textit{stopping}, \textit{stop}(r)\rangle \quad \text{(w20a-w20b)}$$
$$\varphi_\omega(r) = \langle \textit{background}, \textit{word}(r)\rangle \quad \text{(w30a-w30c)}$$
$$\varphi_\omega(r) = \langle \textit{event}(r), \textit{word}(r)\rangle. \quad \text{(w31-w34)}$$

These $\varphi_\omega(r)$ factors will be concatenated with $\varphi_\mu(r)$, as well as those of the scene generation process, to produce the full sequence of factors in the complete grammar as described in the next section.

8.1.3 Scene generation: $P(s,z|m,\sigma)$

The scene is generated in parallel with the words in similar fashion, by walking down the meaning representation and translating it into the scene. However, the relationship to the scene, which in the Frog Stories can be thought of a compact representation of

the union of all possible meaning representations, is simpler in many ways. The key constraint of the scene generation model is that the proposed meaning is contained within the scene with probability one. The remainder of the scene (the *background* scene) is generated in a probabilistic fashion by adding extra roles and children onto the nodes of the meaning representation, choosing their labels randomly.

Like word generation, the process walks down the meaning representation, generating portions of the scene that correspond to each node and edge in the meaning. For each node in the meaning, it and each of its children are added to the scene with probability one, and an additional number of background subtrees are added, determined by a roll of a die with outcomes from zero up to some corpus dependent maximum. These background trees are then generated by drawing the root role label from another multinomial distribution, followed by the child background node label. The number of background edges appearing under background nodes are also determined by rolling a die with outcomes from zero to some corpus determined maximum, and their labels and children are determined by repeating the process recursively.

The rules in Table 8.3 extend the word-generating rules to generate the scene in addition to the meaning and words, adding scene-generating weights $\sigma(r)$ to each rule r. The first element of each of the rule right-hand sides dictates the contribution to the scene of the portion of the meaning representation in the second element of the rule right-hand side. For instance, rule s1a simply duplicates the *look* node and starts the process of generating its *agent* and *theme* child edges. This rule has a scene-generating weight of

$$\begin{aligned}\sigma(s1a) &= P(\text{look}(e) \wedge \text{agent}(e,x) \wedge \text{theme}(e,y) | \text{look}(e) \wedge \text{agent}(e,x) \wedge \text{theme}(e,y), \sigma) \\ &\quad \cdot P(\textit{bg-children} = 0 | \text{look}, \sigma) \\ &= P(\textit{bg-children} = 0 | \text{look}, \sigma)\end{aligned}$$

The rule simply duplicates the node and its children and adds no background subtrees, but other node-generating rules such as s2b do a little more by embellishing the meaning representation with a few more details, in this case by adding one additional child beneath the *boy* node so that $\sigma(s2b) = P(\textit{bg-children} = 1 | \text{boy}, \sigma)$. Edge-generating rules such as s11a through s12 similarly ensure that the thematic roles in the meaning representation are also present in the scene, a process that is completely deterministic.

Table 8.4 lists the rules that govern the generation of the background trees in the scene description. Rule s0 begins the derivation by determining how many additional trees there are in the scene forest, only one in this case with probability

$$\begin{aligned}
\text{N[ROOT]} \rightarrow \langle\ & \lambda e.\text{look}(e) \wedge \text{E[LOOK/}\underline{\text{AGENT}}\text{]}_{\boxed{0}}(e) && \text{(s1a)}\\
& \quad \wedge \text{E[LOOK/}\underline{\text{THEME}}\text{]}_{\boxed{1}}(e) \ \|\ \\
& \lambda e.\text{look}(e) \wedge \text{E[LOOK/}\underline{\text{AGENT}}\text{]}_{\boxed{0}}(e)\\
& \quad \wedge \text{E[LOOK/}\underline{\text{THEME}}\text{]}_{\boxed{1}}(e) \ \|\ \\
& \text{E[LOOK/}\underline{\text{AGENT}}\text{]}_{\boxed{0}}\ \text{W[LOOK]}\ \text{E[LOOK/}\underline{\text{THEME}}\text{]}_{\boxed{1}}\ \rangle\\
\text{N[LOOK/}\underline{\text{AGENT}}\text{]} \rightarrow \langle\ & \lambda x.\text{boy}(x) \wedge \text{E[BG]}(x) \ \|\ && \text{(s2b)}\\
& \lambda x.\text{boy}(x) \ \|\ \text{W[BG]}\ \text{W[BOY]}\ \rangle\\
\text{N[LOOK/}\underline{\text{THEME}}\text{]} \rightarrow \langle\ & \lambda x.\text{frog}(x) \wedge \text{E[FROG/}\underline{\text{LOC}}\text{]}_{\boxed{0}}(x) \ \|\ && \text{(s3b)}\\
& \lambda x.\text{frog}(x) \wedge \text{E[FROG/}\underline{\text{LOC}}\text{]}_{\boxed{0}}(x) \ \|\ \\
& \text{W[BG]}\ \text{W[FROG]}\ \text{E[FROG/}\underline{\text{LOC}}\text{]}_{\boxed{0}}\ \rangle\\
\text{N[FROG/}\underline{\text{LOC}}\text{]} \rightarrow \langle\ & \lambda x.\text{jar}(x) \ \|\ \lambda x.\text{jar}(x) \ \|\ \text{W[BG]}\ \text{W[JAR]}\ \rangle && \text{(s4b)}\\
\hline
\text{E[LOOK/}\underline{\text{AGENT}}\text{]} \rightarrow \langle\ & \lambda e.\text{agent}(e,x) \wedge \text{N[LOOK/1}\underline{\text{AGENT}}\text{]}_{\boxed{0}}(x) \ \|\ && \text{(s11a)}\\
& \lambda e.\text{agent}(e,x) \wedge \text{N[LOOK/}\underline{\text{AGENT}}\text{]}_{\boxed{0}}(x) \ \|\ \\
& \text{N[LOOK/}\underline{\text{AGENT}}\text{]}_{\boxed{0}}\ \rangle\\
\text{E[LOOK/}\underline{\text{THEME}}\text{]} \rightarrow \langle\ & \lambda e.\text{theme}(e,x) \wedge \text{E[LOOK/}\underline{\text{THEME}}\text{]}_{\boxed{0}}(x) \ \|\ && \text{(s11b)}\\
& \lambda e.\text{theme}(e,x) \wedge \text{E[LOOK/}\underline{\text{THEME}}\text{]}_{\boxed{0}}(x) \ \|\ \\
& \text{N[LOOK/}\underline{\text{THEME}}\text{]}_{\boxed{0}}\ \rangle\\
\text{E[FROG/}\underline{\text{LOC}}\text{]} \rightarrow \langle\ & \lambda x_0.\text{loc}(x_0,x_1) \wedge \text{N[FROG/}\underline{\text{LOC}}\text{]}_{\boxed{0}}(x_1) \ \|\ && \text{(s12)}\\
& \lambda x_0.\text{loc}(x_0,x_1) \wedge \text{N[FROG/}\underline{\text{LOC}}\text{]}_{\boxed{0}}(x_1) \ \|\ \\
& \text{N[FROG/}\underline{\text{LOC}}\text{]}_{\boxed{0}}\ \rangle\\
\hline
\text{W[LOOK]} \rightarrow \langle\ & - \ \|\ - \ \|\ \text{look}\ \rangle && \text{(s31)}\\
\text{W[BOY]} \rightarrow \langle\ & - \ \|\ - \ \|\ \text{boy}\ \rangle && \text{(s32)}\\
\text{W[FROG]} \rightarrow \langle\ & - \ \|\ - \ \|\ \text{frog}\ \rangle && \text{(s33)}\\
\text{W[JAR]} \rightarrow \langle\ & - \ \|\ - \ \|\ \text{jar}\ \rangle && \text{(s34)}
\end{aligned}$$

Table 8.3: A grammar fragment for jointly generating the meaning-to-scene and meaning-to-word maps in Figure 8.1. There are three categories: (from the top) foreground event/entity, role label, and word generation rules.

$$\text{N[START]} \rightarrow \langle\ \text{forest}(x) \wedge \text{E[ROOT]}_{\boxed{0}}(x) \wedge \text{E[BG]}(x)\ \Big\|\ \text{E[ROOT]}_{\boxed{0}}(e)\ \Big\|\ \text{E[ROOT]}_{\boxed{0}}\ \rangle \quad \text{(s0)}$$

$$\text{E[ROOT]} \rightarrow \langle\ \lambda e.\text{root}(e,x) \wedge \text{N[ROOT]}_{\boxed{0}}(x)\ \Big\|\ \text{N[ROOT]}_{\boxed{0}}(e)\ \Big\|\ \text{N[ROOT]}_{\boxed{0}}\ \rangle \quad \text{(s10a)}$$

$$\text{N[BG]} \rightarrow \langle\ \lambda s.\text{happy}(s) \wedge \text{E[BG]}(x)\ \Big\|\ -\ \Big\|\ -\ \rangle \quad \text{(s5)}$$

$$\text{N[BG]} \rightarrow \langle\ \lambda x.\text{frog}(x) \wedge \text{E[BG]}(x)\ \Big\|\ -\ \Big\|\ -\ \rangle \quad \text{(s6)}$$

$$\text{E[BG]} \rightarrow \langle\ \lambda s.\text{affect}(s,x) \wedge \text{N[BG]}(x)\ \Big\|\ -\ \Big\|\ -\ \rangle \quad \text{(s10b)}$$

$$\text{E[BG]} \rightarrow \langle\ \lambda s.\text{experiencer}(s,x) \wedge \text{N[BG]}(x)\ \Big\|\ -\ \Big\|\ -\ \rangle \quad \text{(s15)}$$

$$\text{E[BG]} \rightarrow \langle\ \lambda s.\text{loc}(s,x) \wedge \text{N[BG]}(x)\ \Big\|\ -\ \Big\|\ -\ \rangle \quad \text{(s16)}$$

$$\text{W[BG]} \rightarrow \langle\ -\ \Big\|\ -\ \Big\|\ \text{U[BG] W[BG]}\ \rangle \quad \text{(s20a)}$$

$$\text{W[BG]} \rightarrow \langle\ -\ \Big\|\ -\ \Big\|\ \text{U[BG]}\ \rangle \quad \text{(s20b)}$$

$$\text{U[BG]} \rightarrow \langle\ -\ \Big\|\ -\ \Big\|\ \text{the}\ \rangle \quad \text{(s30a)}$$

$$\text{U[BG]} \rightarrow \langle\ -\ \Big\|\ -\ \Big\|\ \text{at}\ \rangle \quad \text{(s30b)}$$

$$\text{U[BG]} \rightarrow \langle\ -\ \Big\|\ -\ \Big\|\ \text{in}\ \rangle \quad \text{(s30c)}$$

Table 8.4: Rules for generating the background event/entities and words for the meaning-to-word map in Figure 8.1. The first pair of rules start the process by selecting a foreground root and demoting all others to the background. The four remaining types of rules, in order, are the foreground event selection, background event/entity, role label, and unigram stopping and word generation rules.

$\sigma(s0) = P(\textit{bg-children} = 1|\textit{root}, \sigma)$. Rules s5-s16 then generate these trees and any subtrees added by the meaning representation duplication rules s1a-s4b. Rules that generate background nodes such as s5 select the node label and choose the number of children with probability

$$\sigma(s5) = P_{bg}(\text{happy}|\sigma)P(\textit{bg-children} = 1|\sigma).$$

Background edges are generated in similar manner by rules s10b-s16 with scene-

generating weights such as

$$\sigma(s16) = P_{bg}(\text{loc}|\sigma).$$

To summarize, let latent variable z be a mapping from meaning representation m onto some isomorphic tree of scene description s, $s/_z m$ be the background portion of the scene (i.e. the scene description minus the that m maps to via z), and, finally, let c_e be the number of children of node e in the background. Then we have the following equation.

$$P(s,z|m,\sigma) = \prod_{e\in m} P(c_e|e,\sigma) \prod_{e\in s/_z m} P_{bg}(e|\sigma)P(c_e|\sigma) \prod_{t\in roles(s/_z m)} P_{bg}(t|\sigma)$$

That is, we add c_e subtrees to each e according to a multinomial distribution over numbers from zero to some corpus-determined maximum, and then generate each background node and edge according to $P_{bg}(e|\sigma)$ and $P_{bg}(t|\sigma)$, respectively.

The scheme can be implemented with four feature functions:

- *event*(r) identifies the type of the event in the meaning portion of rule r,
- *bg-nonterms*(r) returns the count of background nonterminals in scene portion of r,
- *bg-event-nonterms*(r) identifies the event and the count of background nonterminals in the scene portion, and
- *bg-role*(r) which identifies the role type in the scene portion of the rule.

We define the scene portion of the rule factorization function with these feature functions as follows:

$$\varphi_\sigma(r) = \langle \textit{forest}, \textit{bg-nonterms}(r)\rangle \quad \text{(s0)}$$
$$\varphi_\sigma(r) = \langle \textit{event}(r), \textit{bg-nonterms}(r)\rangle \quad \text{(s1a-s4b)}$$
$$\varphi_\sigma(r) = \langle \textit{background-scene}, \textit{bg-event-nonterms}(r)\rangle \quad \text{(s5-s6)}$$
$$\varphi_\sigma(r) = \langle \textit{background-scene}, \textit{bg-role}(r)\rangle \quad \text{(s10b-s16)}$$
$$\varphi_\sigma(r) = \langle \emptyset, \emptyset\rangle. \quad \text{(s11a-s34, s10a, s20a-s30c)}$$

The full set of factors for each rule is the concatenation of those for the meaning, utterance, and scene generating factors:

$$\varphi(r) = \varphi_\mu(r)\varphi_\omega(r)\varphi_\sigma(r).$$

Here, rule s1a inherits its ω and μ weights from rule w1a and rule m1, rule s2b from w2b and m2, s11a from w11a and m11a, and so on. Rules s0 and s10a are deterministic in terms of the meaning- and utterance-generating portions, so we can either define $\varphi_\mu(r) = \varphi_\omega(r) = \langle \emptyset, \emptyset \rangle$ for these two rules or set them to the pair sequences of length zero.

The probabilistic independence assumptions described by the plate diagram in Figure 8.2 follow directly from this definition of the rule factorization function. Because the φ_μ factors are only defined in terms of the meaning portion of the rule, m has no dependencies on either the scene s or words w. Defining the φ_ω in terms of the words and meaning where features of the meaning always appear as the first in feature pairs results in a dependency between m and w but enforces the conditional independence with the scene s. Similarly, the definition of φ_σ is strictly in terms of the scene and the meaning, where, again, all features of the meaning portion of the rule appear as the first element of each feature pair, further enforcing the conditional independence between w and s given m. Without the multi-weighted extension to weighted grammars, such independence assumptions would be difficult to implement, and would require a significant refactoring of the grammar where any such refactoring would almost certainly render our parsing algorithm in Chapter 4 inapplicable.

8.2 Evaluation

We train the model according to the algorithm derived in Chapter 5 and estimate the Dirichlet parameters using Empirical Bayes. The objective of a word learning model is to infer the lexicon that best explains the data, where a lexicon is conventionally represented by a set of concept-word pairs. Our model encodes this information in the weights of the rules, which we tease out by estimating the expected count for instances of each word in the corpus being drawn from foreground distribution associated with e,

$$E[e, word] = E[e]P(word|e, \omega)$$

versus the background distribution

$$E[bg, word] = E[bg]P_{bg}(word|\omega).$$

Here, $E[e]$ and $E[bg]$ are the expected counts of words in an utterance drawn from the foreground distribution of e and the background distribution. These two expectations

can be computed using a variant of the inside-outside algorithm where the derivation forest consists of all possible derivations for meaning representation trees of some maximum depth d and utterances of length n, where we use $d = 4$ and $n = 10$. Using these counts, we extract the lexicon by calculating

$$\textbf{argmax}_e E[e, word]$$

for each word, where e ranges over the set of all possible concept labels plus *bg*. If the maximum is produced by *bg*, the word is classified as a background word and omitted, otherwise pair $\langle e, word \rangle$ is inserted into the lexicon.

Given the inferred set of concept-word pairs and the gold set, we can compute the amount of overlap between them as measured by precision p, recall r, and f-score f. Precision is the proportion of pairs in the inferred lexicon that are also present in the gold, recall is the proportion of pairs in the gold lexicon that are in the inferred set, and f-score is their harmonic mean:

$$f = \frac{2 \cdot p \cdot r}{p + r}.$$

8.2.1 Unstructured scenes

To calibrate our approach, we first compare our model against previous work, shown in Table 8.5. These experiments are performed on the Rollins corpus which, unlike the Frog Stories, lacks structured scene representations. That is, scene descriptions consist of sets of isolated entities and utterance "meanings" are simply subsets of these scene descriptions. In our forest-based scene description conventions, these sets simply translate into a forest of height-zero trees (i.e., tees with childless roots), one per entity in the scene, linked together into a single tree by adding a root node and edges to each of the entity trees.

Performance is comparable to if slightly lower than that reported for Frank et al. (2009), showing that our model, although designed to handle much more structured data, still performs reasonably well without this extra information. In fact, the syntactic features that our model relies on for syntactic bootstrapping are also neutralized since the word order patterns the model learns depend on thematic role labels which are completely absent in Rollins. We now turn to the experiments of main interest on the Frog Stories with structured scene and meaning representations.

	Lexicon		
	p	r	f
association frequency	0.06	0.26	0.10
conditional probability (object—word)	0.07	0.21	0.10
conditional probability (word—object)	0.06	0.32	0.11
mutual information	0.06	0.47	0.11
Yu & Ballard 2007	0.15	0.38	0.22
Frank et al. 2009	0.66	0.47	0.55
this work	0.54	0.33	0.41

Table 8.5: Comparison of word learning models with unstructured scenes on the Rollins corpus. All scores for competing systems are taken from Frank et al. (2009).

8.2.2 The contribution of relational information

Figure 8.3(a) shows a pre-forestized scene description corresponding to the logical expression

$$\begin{aligned}&\text{chase}(e_0) \wedge \text{agent}(e_0, x_1) \wedge \text{bees}(x_1) \wedge \text{theme}(e_0, x_2) \wedge \text{dog}(x_2) \\ \wedge&\text{disturb}(e_1) \wedge \text{agent}(e_1, x_3) \wedge \text{patient}(e_1, x_4) \wedge \text{owl}(x_4) \wedge \text{loc}(e_1, x_5) \\ &\wedge\text{fall}(e_2) \wedge \text{patient}(e_2, x_3) \wedge \text{boy}(x_3) \wedge \text{source}(e_2, x_5) \wedge \text{tree}(x_5).\end{aligned}$$

There are several sources of information, which we can break down into roughly four parts: entity labels, the thematic role linkages (i.e., which concepts share a binary relation), the thematic role labels, and the event type labels. To explore the individual contributions of each, we can experiment by constructing slightly different scene descriptions by removing each type of information, retraining the model, and measuring the quality of the inferred lexicon. Figure 8.3(b)-(c) illustrates three different scene description types based on different combinations of the four sources of information.

- Entities Only (Figure 8.3(d)): At this level of structure, there are no constraints on the subsets of entities that could be talked about. Any subset of entities is equally valid. This scenario approximates the basic setting of most previous computational work in word-learning (Frank et al., 2009; Jones et al., 2010; Johnson et al., 2010; Yu and Ballard, 2007; Yu, 2006; Fazly et al., 2010; Alishahi and Fazly, 2010), and matches the calibration experiments in Section 8.2.1.

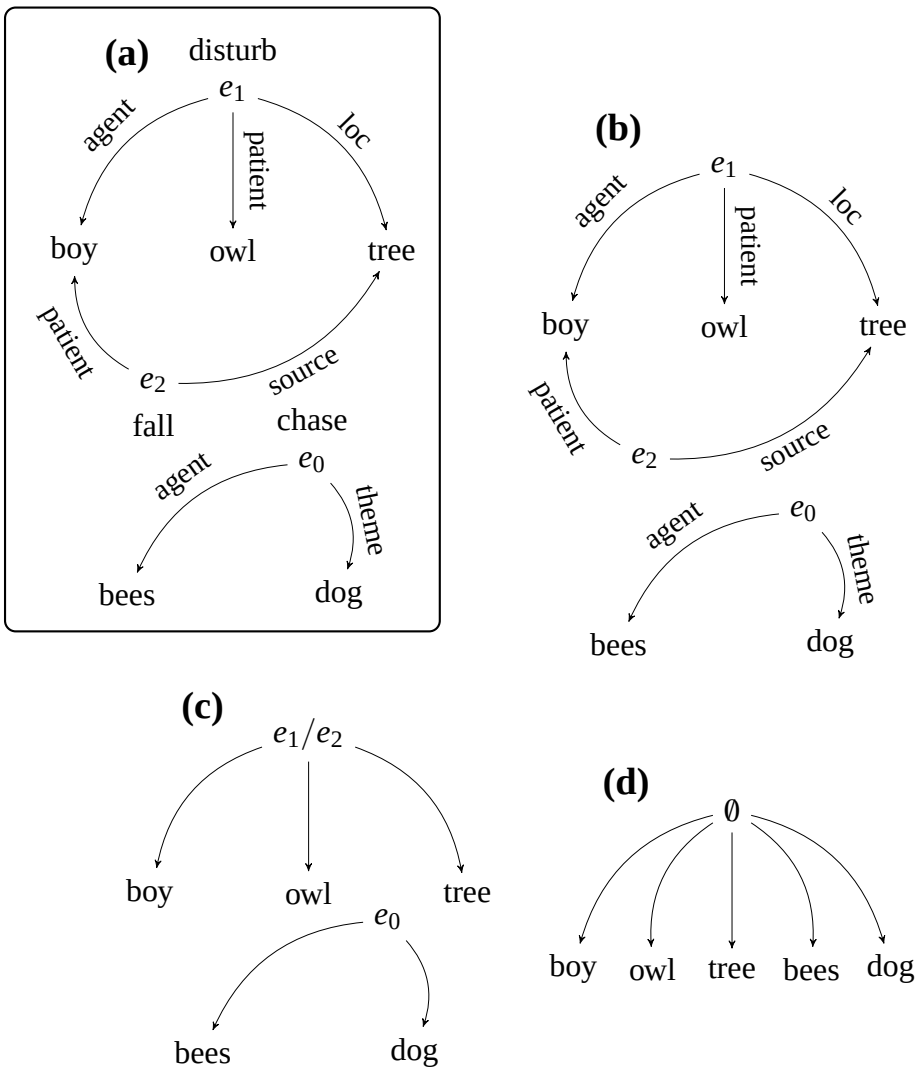

Figure 8.3: A scene description (a) and three variants produced by removing (b) the event type labels, (c) the event and semantic role labels, and (c) everything but the entities.

- Anonymous Relations (Figure 8.3(c)): With the addition of information about which entities interact in the scene, the model has a little more help, since one might reasonably assume that entities that interact in the scene are more likely to be discussed together in the same utterance. Such an assumption constrains the entity subsets, even if the learner has no way of knowing what the exact nature of the interaction are or what role each entity plays in it. Observe that in the scene in the figure, $\{$*tree, owl*$\}$ and $\{$*tree, owl, boy*$\}$ are both eliminated as possible subsets since there is no direct interaction between these entities,

helping to narrow the search space and aid learning.

- Anonymous Relations and Events (Figure 8.3(b)): Additional information about the roles of the entities – which entity is acting and which one is being acted upon – further helps constrain the relationship between world and words. For instance, the learner can take advantage of English's tendency to place the *agent* at the front and the *theme* at the end of a sentence to guess that the word for *dog* is likely to appear at the end of sentences given the scene in the figure.

We are primarily interested in the contribution of relational information to the learning problem, the question being whether this level of structure in the scene and meaning representations coupled with joint syntactic learning have a significant impact on learning. For our purposes, then, we are interested in levels (b), (c), and (d). In the entities-only scenario (d), meanings are completely unstructured and scenes contain no information about which entities are likely to co-occur in an utterance, so word learning models would have a difficult time eliminating most subsets of entities, a priori, and the number of possible subsets is exponential in the size of the scene description. In the example shown in Figure 8.3(d), there are 5 entities, leading to $2^5 = 32$ meaning candidates. This may not seem like such a large number (in fact, it is about what word learners operating on the Rollins corpus must handle) but this toy example is much smaller than any of the scenes in the Frog Stories corpus where there can be up to 20 entities, leading to $2^{20} \approx 1,000,000$ possible candidates, thousands of times the number for previous work. This number is daunting even for our grammar-based model which relies on dynamic programming for inference. In contrast, the scene description with anonymous events and roles shown in (b) is much less costly since, restricting meanings to include at most one tree in the scene forest, there are only $2^3 + 2^2 - 1 = 11$ meaning candidates to explore in our toy example, and the Frog Stories corpus has up to about two hundred candidates for some scenes at this level of structure. We thus restrict ourselves to experiments involving the two more structured scenarios involving anonymous roles and events. By contrasting performance between scenarios (b) and (c), scene descriptions with and without role labels, we can isolate the contribution of role labels and explore the benefit of learning the order in which they tend to appear within an utterance.

8.2.2.1 Further constraining information

Even after exploiting structural information to vastly narrow the search space, there remain over 300 meaning candidates for some scenes, posing a challenge for learning. Even parsing is difficult since the parse forests are very large, requiring memory on the order of $O(|s| \cdot |w|^2 \cdot G)$, where G is the grammar constant which, in the worst case, can be exponential in the grammar. This computational expense motivates the incorporation of several aids to prune the search space. In particular, we rely on four types of constraints

- prior knowledge in the form of a small seed lexicon of concept-word pairs,
- linguistic salience to distinguish content words from function words,
- non-linguistic salience to focus attention on the most central concepts of the story,
- structural constraints on the meaning candidates such as connectedness and completeness.

The last two, non-linguistic salience and structural constraints, are as described in Section 7.5 and have already been incorporated into the scene forests for our purposes (and the 300 meaning candidates mentioned assumes these two types of constraints have already been taken into account).

As for prior knowledge, we aim to simulate word learners who are somewhat farther along in the process than those of (Frank et al., 2009), therefore assume the learner has already acquired a small set of seed words. In fact, our data set of roughly 1000 utterances per language amounts to about 20-30 minutes of input, and seems better suited for simulating how a learner might perform at a certain stage of development rather than a full study of development from infancy to adulthood. In particular, psychologists tend to agree that syntactic bootstrapping effects are more likely to show after at least a few words have already been acquired. Furthermore, starting with a small set of known words may help to better highlight the potential impact of syntax in the resolution of referential ambiguity for the remaining words. Thus, for our simulations, we assume the model already knows the words for the three most frequent entities in the corpus: dog, boy, and frog.

It is common practice to use such seed lexicons in semantic parsing applications, where systems are often fed a list of named entities, effectively giving the model a gold

lexicon for the entire set of entities in the corpus. Given that these entity-word pairs are precisely what the model is intended to learn in our setting, however, this is clearly not appropriate for our purposes, but we can still explore the effect of joint syntactic learning with a small, partial seed lexicon. This assumption probably better approximates the situation of most human word learners who are at the stage of learning the syntax of their language. In terms of computational complexity, this small seed lexicon helps to eliminate many false analyses; when the model encounters a sentence with a known word (e.g., "dog"), it can safely rule out meaning candidates that exclude the *dog* entity. Similarly, if the model does not observe any words for the known concepts in the utterance, it can rule out meaning candidates that *include* those concepts.

It has also been argued that acoustic features of speech can serve to heighten the salience of content words, thus drawing a distinction between content and function words. Fernal and Mazzie (1991), for instance, observed that content words are more likely to occur at intonational peaks, setting them apart from function words. This situation can be approximated in our model by using a stop list of function words to relegate them to the background word distribution. The stop list can serve to drive down referential ambiguity by eliminating a need to explore word-meaning maps (and the corresponding parses) that incorrectly link function words to entities and event predicates.

8.2.2.2 Human-subject-esque testing

Evaluation on the Frog Stories is similar to that for the word learning scenario of prior work already described but with a slight departure. In particular, in a procedure more similar to human experiments (Yu and Smith, 2007; Smith et al., 2011; Graf Estes and Hurley, 2013), testing the model consists of presenting the set of nouns in the lexicon one by one and asking for the model's best guess for the object meaning of each. This procedure differs subtly from that of Frank et al. (2009) where many words that are excluded from the lexicon (e.g., verbs or function words) are also presented at test time (where the gold proposal is the null object). Thus, our test procedure, while more similar to that used in human experiments, includes fewer distractors and is thus somewhat easier than the test procedure employed with previous computational models. Our primary motivation for this choice is due to the fact that learning is much harder in our case than previous work, with far *more* distractors at training time, and subtle changes in performance are more difficult to measure with the same level of test-time noise.

scenario	English	German	Turkish
anonymous roles	0.29	0.14	0.08
anonymous events	0.58	0.23	0.07

Table 8.6: Learning words for entities given different amounts of information in the scene. *Anonymous roles* corresponds to Figure 8.3(c), including only the entity labels and information about which ones are related to each other, while anonymous events corresponds to Figure 8.3(b) where type labels for the roles the entities play in these interactions have been added. Event type labels themselves are omitted in both cases to reduce the search space and focus on the contribution of role labels.

8.2.2.3 Performance

Table 8.6 shows the performance of the model while employing these additional pieces of information across our two layers of structure and three different languages. As we can see by the considerable difference in performance for anonymous roles and anonymous events there is a sizable performance boost when we add the binary relation labels into the scene representation. The boost is mainly due to syntactic bootstrapping, since the primary thing the model learns with the binary relation labels included is the typical order such relations are likely to occur in the sentence (i.e., agent-event-theme type ordering information).

Performance on English is considerably higher than for the other two, which is somewhat to be expected given the implicit bias in the annotations toward English vocabulary. Additionally, both German and Turkish are far more agglutinative, and Turkish in particular has several inflections for each verb and noun, reflecting case marking and so on, leading to a sparsity problem and suggesting that we might need to include a morphological element to model Turkish word learning. Related to morphology, the Turkish sentences also include a greater degree of variability in the ordering of thematic roles in the utterance, partly due to the offloading of information from word order to morphology and partly because morphological alternations can signal different standard orders for verb arguments. The German data is also difficult because utterances tend to be longer, leading to greater referential ambiguity. However, we do see a similar trend in German to the English performance.

It is also interesting to explore the effect of the seed lexicon on performance. Hav-

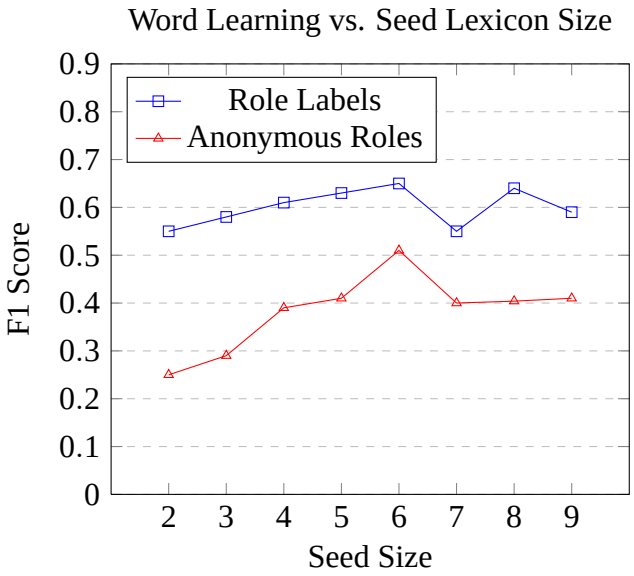

Figure 8.4: The effect of seed lexicon size on word learning performance for the cases where the model can and cannot learn canonical orders in which thematic roles are realized in utterances. Words in the seed are excluded from the test set, and numbers are for the English section of Frog Stories.

ing a few already known words can boost a learner's productivity in picking out new ones, but how many does a learner need to know for the effect to show? Figure 8.4 plots the change in performance for unknown concepts as concepts are added to the seed lexicon, contrasting the behavior of the role order learning model vs. the model learned with anonymous roles. Interestingly, the first few concepts make the biggest difference after which there are diminishing returns, leveling off at around six seed concepts or so. This is partly due to the fact that we have added concepts in order of decreasing frequency in the corpus, so the first two concepts are the most frequent. After all, the first few concepts, which are the most useful for learning other concepts because they are most frequent and often also occur in the greatest number of contexts, allow the model to learn many of the less frequent concepts that are added to the right in the graph in Figure 8.4. However, the role order learning model peaks much faster, apparently making more efficient use of the first two or three concepts in the seed lexicon. This suggests that even a very small arsenal of words can make a big difference for a learner, especially for those capable of identifying and exploiting word order patterns. Of course, it depends on the particular corpus which words will be most useful,

but overall, assuming words are learned roughly in order of decreasing frequency, one might expect a similar effect. Essentially, in the world of the Frog Stories, all events revolve around either the boy, the dog, or the frog, so knowing the words for these characters gives us most of what we could hope to gain from a seed lexicon. However, in the real world, the vocabulary and set of concepts to be learned are both much larger than can be represented by a fairly small corpus like the Frog Stories, and these curves would likely level off more slowly for larger corpora.

8.3 Discussion and conclusion

Our primary interest is twofold: (1) to examine the implicit assumption of previous behavioral and computational studies that learners must somehow narrow down the set of candidate meanings to a handful of possibilities for effective learning, and (2) to test the effect of syntactic bootstrapping on learning in highly ambiguous settings. While we have employed several simplifying assumptions to make the problem more tractable, we have explored these questions with roughly thirty times the amount of ambiguity reported for previous semantic parsing-based word learners in terms of the number of meaning candidates per sentence.

It is mainly due to the efficiency of our particular choice of modeling framework that allows us to explore this larger space of potential meaning candidates per utterance. Let $|s|$ be the size of the scene graph after being converted to a forest as described in Section 7.4. Then, using the synchronous parsing algorithm described in Section 4.4 to jointly parse scene and utterance w, the time complexity for a single utterance in our approach is $O(|s| \cdot |w|^3)$. This is in contrast to other approaches such as that of Chen and Mooney (2008) which have a time complexity of $O(|m| \cdot |w|^3)$ per meaning candidate m, of which there are essentially as many as there are subtrees in the trees of forest s, leading to a worst case bound of $O(2^{|s|})$ possible meaning candidates. Thus, for them, parsing takes time exponential in the size of the scene, and their overall parsing bound is $O(2^{|s|} \cdot |m| \cdot |w|^3)$ per utterance where $|m|$ is the size of the largest meaning candidate. The approach of Kwiatkowski et al. (2013) has a similar analysis but is somewhat more expensive due to the greater flexibility allowed in the meaning-to-words map. Our purely grammar-based model permits us to take full advantage of the dynamic programming solutions to parsing and inference, so we can share common sub-analyses and avoid redundant recomputation across different meaning candidates with shared sub-structure, exploring all $O(2^{|s|})$ candidates at once in time linear in s.

While it is sometimes assumed that arguments in favor of more powerful learning mechanisms and arguments in favor of learning biases are at odds, we believe in the importance of both. In fact, our work can be seen as decidedly on the side of the importance of such learning constraints. At the same time, our corpus study in Chapter 7 suggests that the effect of these constraints may be somewhat overrepresented, as even after the application of several highly constraining assumption the task remains far from solved, with a level of referential ambiguity that far exceeds previously explored computational scenarios. Specifically, we have relied on notions of scene-level saliency to reduce the complexity of the scene description, and granted the model additional knowledge in the form of a small seed lexicon and the distinction between function and content words. However, even at the reduced level of ambiguity after scene-level pruning, the roughly 200 meaning candidates with anonymized events that our model faces, while far less than the unpruned, unanonymized 1,300 candidates, far exceeds previous work that has been limited to a few dozen in the simple unstructured scenario of work such as Frank et al. (2009) or just a handful in the structured case such as that of Kwiatkowski et al. (2012). Thus, we are in the position to test synergistic effects between learning syntax and word meanings in a fairly different setting than previously possible.

In our preliminary explorations of this expanded space, we find that the model is still capable of learning even under scenarios of greater ambiguity, setting the stage for testing various learning effects. As an example of such and effect, we observe that the model performs better when it is given sufficient information to learn some simple facts about canonical word order (i.e., the order in which thematic roles tend to be realized), demonstrating that joint syntactic learning further improves learning performance. It remains to be seen whether humans are able to learn as well as our statistical model under similar circumstances, but the success of our model suggests that a failure to learn would likely be due more to computational rather than data limitations.

Finally, we also find that the importance of structural information in the scene description proves essential for managing computational overhead. Flat scenes consisting only of a set of disconnected entities found in most previous computational work in word learning such as Frank et al. (2009) or Fazly et al. (2010) offer no information about which entities are likely to be discussed in the same utterance, but relational information allows a learner to better focus on the more plausible subsets. It helps to focus on sets of entities that relate to each other in the scene, since these sets are also more likely to co-occur in an utterance. Are there other types of structural information

that might further aid in learning?

Chapter 9

Conclusion

We set out to explore how relations between words and relations between entities in the world might interact in the acquisition of word meanings and in the process produced several technical innovations, a corpus, and two different probabilistic models, one for semantic parsing, the other for word learning. Observing that in some ways semantic parsers and models of word learning are attacking different aspects of the same problem, we introduced a framework under which it is possible to unify the two into a single joint model. To ease our way, This particular choice of model class brings with it many useful tools from context-free grammar, such as the inside-outside algorithm for computing expectations and an efficient method for inferring the most probable parse. We add to this toolbox by introducing three technical innovations: an extension to probabilistic grammar that allows us to express a much larger set of possible models in the same basic framework, a Bayesian inference algorithm for these grammars, and a parsing algorithm for parsing the restricted class of HRG we use for modeling scene and meaning representation languages. Using this toolbox, we develop a semantic parsing model which is modular and extensible, which we modify for a novel word learning model that enjoys greater computational efficiency than other models and can explore a larger hypothesis space, permitting us to test the effectiveness of statistical learning in the face of a large degree of referential ambiguity. Finally, for experiments with this model, we introduce a new data set, annotating the Frog Stories corpus (Berman and Slobin, 1994) with meaning representations for each utterance and scene descriptions to represent non-linguistic context in word learning.

Testing the word learning model necessitates the annotation of a new corpus with scene descriptions and utterance meaning representations in the form of the Frog Stories corpus, work that is described in Chapter 7. The corpus allows the simulation of

word learners who must somehow infer the referents for individual words given only two pieces of information: the words themselves, and the non-linguistic information containing the set of possible referents. The choice to represent non-linguistic context as a single structured description allows experiments on the contribution of such structure to learning. Furthermore, the corpus is naturally segmented by pictures which contain hundreds of elements, leading to a significant amount of uncertainty, with over a thousand meaning candidates to sort through for some utterances. This vast ambiguity sets a much higher bar for future work in computational modeling of word learning, since previous models have tended to focus on far simpler scenarios, contending with orders of magnitude fewer meaning candidates. In fact, our own work only makes a modest start at the problem, leaving much room for improvement in future efforts.

Besides computational simulations, it is also possible that the corpus could serve in other capacities. Our very preliminary analysis in Chapter 7, for instance, already demonstrates that there is far more ambiguity than is typically discussed or assumed in experiments. Of course, this ambiguity is completely dependent on the assumptions one makes on the availability of different types of information, but the corpus may also serve as a useful discussion point here too; by varying one's assumptions and applying them to the corpus, one can explore their effects in terms of reduction in ambiguity. By approximating saliency with frequency information, we demonstrate one example of how a highly complex scenario of over 1,300 meaning candidates can be curtailed to 350 or so. As a multilingual corpus, it is also possible to test models and hypotheses over several languages, invaluable if one is interested in the fundamental learning problem facing children of any linguistic setting as opposed to language specific solutions.

The handling of this large space of meaning candidates motivates a few technical innovations.

- First, we introduced a generalization of probabilistic context-free grammar in Section 3.4 which associates multiple weights per rule in the form of a mini-Bayes net, permitting us to implement a much larger class of probabilistic models in a weighted grammar. In particular, these multi-weighted grammars give us the ability to make additional independence assumptions, allowing one to implement models that are more robust in the face of data sparsity, something that can often be especially problematic with synchronous grammar which, in practice, often have very large rules.

- Second, we derived a variational algorithm in Chapter 5 for performing Bayesian inference with these generalized probabilistic grammars. The algorithm is a strict generalization of that for PCFGs, owing to the fact that the underlying model is itself a generalization of PCFG. Here too, we enjoy the benefits of building from a standard formalism since we are able to leverage the inside-outside algorithm for key portions of the algorithm.

- Finally, our third technical contribution comes in the form of a novel parsing algorithm for unordered trees. Parsing takes time linear in the size of the tree being parsed, which allows us to model the commutative property of predicate calculus while keeping computational expense at a manageable level. At the same time, while the parser is designed for a special class of graph grammar, the key optimizations are also applicable to general HRG parsing.

Each of these three innovations centers on a fairly broad formalism, giving our contributions a generality as broad as the set of applications one might find for the corresponding model class. Synchronous grammar, for instance, is not only used in semantic parsing, but is often used for syntax-driven machine translation, among other applications. Perhaps the ability to enforce finer grained independence assumptions of multi-weighted grammar could be useful in the large rules required for translating syntactic patterns with medium range dependencies.

There are also close relationships between these formalisms to other classes, where our multi-weighted grammars are generalizations of PCFGs, and the unordered tree grammars are a special case of HRG, which help to situate our contributions in a larger context. One benefit of this larger context is that it may aid in the extension of our algorithms to other formalisms and still further applications. Furthermore, inference algorithms that apply to a general *model class* rather than a specific *model* allow one to focus on designing, extending, and testing the model itself rather than on one-off algorithm design and implementation; so long as each model variation remains in the same class, one can utilize the same algorithm and software package over and over again.

The semantic parsing and word learning models in Chapter 6 and Chapter 8 serve as demonstrations of the power of the framework. The semantic parser can be seen as a re-implementation of the Lu et al. (2008) as a synchronous grammar. However, while Lu et al. (2008) developed model-specific parsing and inference procedures, we are able to rely on general grammar-based algorithms. As a result, although our

model is very similar, our framework better supports modular design and makes it far easier to extend the semantic parser, a feature we exploit when designing the word learning case. Furthermore, in terms of performance, our model suffers not at all for exchanging the model-specific algorithms of Lu et al. (2008) for the grammar-general training algorithm of Chapter 5, performing as well or better, and parsing has the same $O(|m| \cdot |w|^3)$ time complexity bound.

In fact, implementation of the word learner in Chapter 6 requires only a few modifications of the semantic parser, which can be seen as a sub-module of the larger word learning model. We merely add one additional monolingual grammar for the scene, integrate this grammar into the semantic parsing grammar, and slightly re-factor the model probabilities to accommodate the increased variability in the Frog Stories data set. The flexibility of the framework should also make future work easier as we explore other model variations. Capturing the entire scene-meaning-word relationship as a synchronous grammar has design implications, since we can use the same standard parsing and inference algorithms as we explore different model variations and extensions. At the same time, our grammar-based approach also has important implications for the computational overhead since we can quickly and compactly enumerate the entire set of possible meaning-to-word mappings as a parse forest computed in $O(|s|^2 \cdot |w|^3)$ time, considerable savings over the bound of $O(2^{|s|} \cdot |m| \cdot |w|^3)$ for the next fastest solution.

In some sense, the word learner is simply a preliminary demonstration, and there are many alternative factorizations and subtly different grammars that might better integrate syntactic or other features. For instance, our model incorporate mutual exclusivity through the use of a Dirichlet prior which promotes sparsity (i.e., a small lexicon), cross-situational learning in the form of a data-driven statistical learner, and syntactic features in the form of ordering patterns. We also incorporate notions of salience by pruning the scene to only the most salient concepts and prosodic cues, a variety of social cue, that serve to draw attention to content words by distinguishing function words. The added efficiency of our framework allows us to test the impact of these features under much higher levels of referential ambiguity than previously explored. This added efficiency has real impact on the set of experiments we can run and the questions we can explore, permitting us to test hypotheses about the fundamental limitations of statistical learning and the contributions of different types of features and biases to word learning. In particular, the results of Chapter 8 demonstrate that a statistical learner can indeed acquire a kind of lexicon under much more ambiguous

circumstances than previously explored. Both behavioral studies and computational work have primarily focused on settings where there were only a handful of possible candidate meanings, while the referential ambiguity in our experiments is orders of magnitude greater but the model still succeeds in inferring a lexicon with several reasonable word-concept associations, albeit with the aid of several simplifying assumptions.

Representing scenes as structured objects rather than sets of completely unrelated concepts also allows us to explore the impact of different forms of information in the scene description. At one extreme, we find that the use of "bag of entities" representations for non-linguistic context, a common sort of scene description in previous computational work (Frank et al., 2009; Fazly et al., 2010; Yu and Ballard, 2007; Jones et al., 2010), provides too few constraints on the types of meaning representations a model must explore. Even our model finds it prohibitive to explore the exponential space of all possible subsets, but a more structured representation of the scene permits the model to make more intelligent guesses, narrowing the search space to something that is far more tractable, if often still challenging. Just adding unlabeled edges to indicate which entities interact makes things much more tractable, something that may seem obvious in retrospect, but one often expects that, to the contrary, richer representations mean greater complexity. In a sense this is true in that processing structured scene descriptions does require extra machinery, but it also proffers extra signposts to guide inference, leading to greater efficiency.

Besides making inference tractable, additional information can also aid learning, even under highly ambiguous settings. In testing the impact of one simple type of syntactic information, specifically, the canonical ordering of thematic roles in a sentence, we found that learning further improves, beyond merely being tractable. Indeed, at least in the case of one language in our experiments, this relatively crude syntactic information about whether agents come before or after themes in a sentence has a significant impact on performance. So, while psychologists have argued that this type of information may be necessary, we find that it does indeed help, but is not necessary at least for a certain basic level of performance. However, the model incorporates no features that allow it to learn about function words or morpho-syntactic properties, leaving the door open for future work. We anticipate adding such extensions might be easier working within our or some other grammar-based framework.

In fact, there are several fruitful avenues open for future work. One could explore alternative sources of information and extensions to the word learner, for one. One

could extend the unordered tree parsing algorithm to handle more general classes of graph. One could continue in the vein of Chapter 6 and assimilate other semantic parsing models into a more general framework for the purposes of drawing new connections among seemingly isolated efforts.

Discourse structure is one interesting direction to take the word learning model. Luong et al. (2013), for instance, utilizes a grammar-based model to parse entire documents in order to exploit discourse-level structure in word learning. However, Luong et al. work with the more typical "bag of entities" style unstructured scene representations. Our framework coupled with structured scene representations make a grammar-based approach to discourse modeling particularly appealing. At present, our model treats every utterance independently, drawing each at random from among the set of statements that are logically entailed by the scene description. However, these utterances form a narrative and are not truly independent. Given that a storyteller already mentioned a particular event, for example, it seems less likely that he will repeat the same statement over again rather than choose some other utterance that would better advance the story. Importantly, the scene description actually encodes the full narrative the storyteller intends to relate, suggesting that there is a more interesting model for breaking this description down into a sequence of smaller statements. Although we had pruned pronouns from the scene descriptions for the sake of simplicity and because discourse was outside our focus, the full, unpruned scenes not only contain pronouns but also indicate to which entities they refer, potentially raising other possibility relating to discourse and coreference.

As an extension to the grammar framework itself, Chapter 4 mentions several ways that our tree parsing algorithm generalizes to the unrestricted HRG parsing problem. Implementing these optimizations in a general HRG parser is an obvious direction to go for future work, particularly because processing general graphs rather than forest approximations could have an important impact on learning. To see how, recall that the forestization procedure in Section 7.4 removes reentrancies by duplicating portions of the graph that are shared between different trees. This means that certain entities like *boy* in the example illustrated in Figure 7.4 are over-represented in the scene. One consequence of this is that, since the model essentially draws subtrees at random, *boy* has multiple chances of being drawn while other entities such as *owl* have only one, biasing the model in favor of meanings involving *boy* over those with *owl*, and therefore preferring to match words with *boy* in the lexicon. Operating from the unforestized graph would eliminate this particular source of bias, possibly improving word learning

performance and reducing the number of false positives in the inferred lexicon.

However, our analysis in Chapter 4 argues that, even after applying our compact symmetric parse encoding scheme, the lack of ordering in the graph being parsed still leads to an algorithm that is exponential in the size of the right-hand side of the grammar rules. Thus, we may wish to consider other classes of graph languages where there is an ordering constraint among the edges. While adopting and ordering would remove the particular feature we chose HRG for, modeling the commutative property, there could be considerable utility in other applications for ordered graphs. Just as an example, semantic dependency graphs have an ordering property imposed by the words of the sentence.

Considering more general graph parsing, it also seems likely that the CCG-based approach of Kwiatkowski et al. (2010) may very well be re-interpreted as a HRG-based model. When one take a closer look at Algorithm 1, it actually closely resembles a synchronous parsing algorithm where the *split* operation that enumerates the various decompositions of the meaning representation essentially does the job of a monolingual HRG parser. This HRG parser would necessarily be more general than the tree parser of Chapter 4, but treating the meaning representation as a graph, each split returned is essentially a particular parse item under some more general graph grammar. In fact, one could think of these splits as a kind of translation grammar somewhat similar to the example grammar given in Table 3.2. The model's reliance on a context-free restriction of CCG makes it especially tempting to attempt to simulate Kwiatkowski et al.'s model with a monolingual HRG for decomposing the meaning into a synchronous grammar that maps these meaning fragments to segments of the string. Kwiatkowski et al. found it necessary to restrict the set of decompositions in certain ways for efficiency reasons, just as we have had to in our own experiments. However, viewing the *split* operation as a HRG parser may suggest alternative ways of restricting the space of decompositions that preserve efficiency while allowing other types of word meanings. In any event, such a project might at least help situate that model in the space of other approaches, perhaps offering other insights.

We have outlined just a few possibilities. There are several ways in which the grammar framework could be extended. There are also many possibilities for extending the semantic parsing and word learning models, perhaps increasing capabilities or allowing one to explore the effect of alternative sources of information or learning biases and their interactions.

Appendix A

The expectation of the log of a multinomial weight under a Dirichlet

The standard form of a distribution in the exponential family is as follows:

$$p(\theta|\alpha) = h(\theta)\exp(\sum_{i=1}^{n}\eta_i(\alpha)t_i(\theta) - g(\alpha)).$$

where θ and α are n-dimensional vectors such that θ is a random variable and α is a parameter vector, and the $t_i(\theta)$ are the sufficient statistics of the distribution. A standard result of the exponential family states that the expectation of a sufficient statistic can be derived by taking the derivative of the log-partition function $g(\alpha)$:

$$E[t_i(x)] = \frac{dg(\alpha)}{d\eta_i(\alpha)}.$$

Restating the Dirichlet in this same standard form results in

$$p(\theta|\alpha) = \exp\left(\sum_{i=1}^{n}(\alpha_i - 1)\ln(\theta_i) - \ln\frac{\prod_{i=1}^{n}\Gamma(\alpha_i}{\Gamma(\sum_{i=1}^{n}\alpha_i}\right).$$

From this, we see that

$$\begin{aligned} h(\theta) &= 1 \\ \eta_i(\alpha) &= \alpha_i - 1 \\ t_i(\theta) &= \ln\theta_i \\ g(\alpha) &= \ln\frac{\prod_{i=1}^{n}\Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n}\alpha_i}, \end{aligned}$$

leading to

$$\begin{aligned} E_{p(\theta|\alpha)}[\ln\theta_i] &= \frac{d}{d\alpha_i}\left[\sum_{i=1}^{n}\ln\Gamma(\alpha_i) - \ln\Gamma(\sum_{i=1}^{n}\alpha_i)\right] \\ &= \Psi(\alpha_i) - \Psi(\sum_{i=1}^{n}\alpha_i). \end{aligned}$$

Bibliography

Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. (2008). Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014.

Akhtar, N. and Montague, L. (1999). Early lexical acquisition: The role of cross-situational learning. *First Language*, 19(57 Pt 3):347–358.

Alishahi, A. and Fazly, A. (2010). Integrating syntactic knowledge into a model of cross-situational word learning. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*.

Artzi, Y. and Zettlemoyer, L. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

Baldwin, D. (1993). Early referential understanding: Infant's ability to recognize referential acts for what they are. *Developmental Psychology*, 29:832–843.

Baldwin, D. A., Markman, E. M., Bill, B., Desjardins, R. N., Irwin, J. M., and Tidball, G. (1996). Infant's reliance on a social criterion for establishing word-object relations. *Child Development*, 67:3135–3153.

Barton, G. E. (1985). On the complexity of id/lp parsing 1. *Computational Linguistics*, 11(4):205–218.

Beal, M. J. (2003). *Variational algorithms for approximate Bayesian inference*. PhD thesis, Gatsby Computational Neuroscience unit, University College London.

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Berg-Kirkpatrick, T., Bouchard-Côté, A., DeNero, J., and Klein, D. (2010). Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590.

Berman, R. A. and Slobin, D. I. (1994). *Relating events in narrative: A crosslinguistic developmental study*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blei, D. (2004). *Probabilistic Models of Text and Images*. PhD thesis, Division of Computer Science, U.C. Berkeley.

Bloom, P. (2000). *How children learn the meanings of words*. MIT Press, Cambridge, MA.

Börschinger, B., Jones, B. K., and Johnson, M. (2011). Reducing grounded learning tasks to grammatical inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Branavan, S. R. K., Silver, D., and Barzilay, R. (2011). Learning to win by reading manuals in a Monte-Carlo framework. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 268–277.

Branavan, S. R. K., Zettlemoyer, L. S., and Barzilay, R. (2010). Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1268–1277.

Braun, M. and McAuliffe, J. (2010). Variational inference for large-scale models of discrete choice. *Journal of the American Statistical Association*, 105(489).

Braune, F., Bauer, D., and Knight, K. (2014). Mapping between English strings and reentrant semantic graphs. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Bresnan, J. (2001). *Lexical functional syntax*. Blackwell.

Brown, P. (1998). Children's first verbs in Tzeltal: Evidence for an early verb category. *Linguistics*, 36:713–753.

Brown, P. F., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Cai, Q. and Yates, A. (2013). Large-scale semantic parsing via schema matching and lexicon extension. In *ACL (1)*, pages 423–433.

Carey, S. (1978). The child as word learner. *Linguistic theory and psychological reality*, pages 264–293.

Chen, D. L., Kim, J., and Mooney, R. J. (2010). Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37:397–435.

Chen, D. L. and Mooney, R. J. (2008). Learning to sportscast: A test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*.

Chen, D. L. and Mooney, R. J. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, pages 859–865.

Chiang, D., Andreas, J., Bauer, D., Hermann, K. M., Jones, B., and Knight, K. (2013). Parsing graphs with Hyperedge Replacement Grammars. In *Proceedings of the annual meeting of the Association for Computational Linguistics*.

Chiang, D., Graehl, J., Knight, K., Pauls, A., and Ravi, S. (2010). Bayesian inference for finite-state transducers. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 447–455.

Choi, S. and Gopnik, A. (1995). Early acquisition of verbs in korean: A cross-linguistic study. *Journal of Child Language*, 22:497–529.

Chomsky, N. (1957). *Syntactic Structures*. Mouton, Paris.

Chomsky, N. (1980). *Rules and representations*. Basil Blackwell, Oxford.

Clarke, J., Goldwasser, D., Chang, M.-W., and Roth, D. (2010). Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 18–27.

Colunga, E. and Smith, L. B. (2005). From the lexicon to expectations about kinds: A role for associative learning. *Psychological Review*, 112:347–382.

Cooper, R. H. (1975). *Montague's semantic theory and transformational syntax*. PhD thesis, University of Massachusetts at Amherst, Amherst, Massachusetts.

Cooper, R. P. and Aslin, R. N. (1990). Preference for infant-directed speech in the first month after birth. *Child Development*, 61(5):1584–1595.

DeNeefe, S. and Knight, K. (2009). Synchronous tree adjoining machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 727–736.

Doe, R. (2014). The Freebase API.

Drewes, F., Habel, A., and Kreowski, H.-J. (1997). Hyperedge replacement graph grammars. *Handbook of Graph Grammars and Computing by Graph Transformation*, pages 95–1626.

Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Engelfriet, J. (1977). Top-down tree transducers with regular look-ahead. *Mathematical Systems Theory*, 10:289–303.

Fazly, A., Alishahi, A., and Stevenson, S. (2010). A probabilistic computational model of cross situational learning. *Cognitive Science*, 34(6):1017–1063.

Feder, J. (1971). Plex languages. *Information Sciences*, 3:225–241.

Fernal, A. and Mazzie, C. (1991). Prosody and focus in speech to infants and adults. *Developmental Psychology*, 27(2):209–221.

Fernald, A. (1985). Four-month-old infants prefer to listen to motherese. *Infant Behavior and Development*, 8:181–195.

Flanigan, J., Thomson, S., Carbonell, J., Dyer, C., and Smith, A. N. (2014). A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1426–1436.

Fleischman, M. and Roy, D. (2005). Intentional context in situated natural language learning. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 104–111.

Frank, M. C., Goodman, N. D., and Tenenbaum, J. B. (2009). Using speakers' referential intentions to model early cross-situational word learning. *Psychological Science*, 20:579–585.

Galley, M., Hopkins, M., Knight, K., and Marcu, D. (2004). What's in a translation rule? In *Proceedings of the annual meeting of the North American Association for Computational Linguistics*.

Gasser, M. and Smith, L. B. (1998). Learning nouns and adjectives: A connectionist approach. *Language and Cognitive Processes*, 13:269–306.

Ge, R. and Mooney, R. J. (2005). A statistical semantic parser that integrates syntax and semantics. In *Proceedings of CoNLL-2005*.

Gentner, D. (1982). Why nouns are learned before verbs: Linguistic relativity versus natural partitioning. *Language Development*, 2:62–73.

Gillette, J., Gleitman, H., Gleitman, L., and Lederer, A. (1999). Human simulations of vocabulary learning. *Cognition*, 73(2):135–176.

Gleitman, L. (1990). The structural sources of verb meanings. *Language Acquisition*, 1:135–176.

Goldwasser, D., Reichart, R., Clarke, J., and Roth, D. (2011). Confidence driven unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 1486–1495.

Goldwater, S., Griffiths, T. L., and Johnson, M. (2009). A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Graf Estes, K. and Hurley, K. (2013). Infant-directed prosody helps infants map sounds to meanings. *Infancy*, 18(5):797–824.

Grimshaw, J. (1981). Form, function,and the language acquisition device. *The logical problem of language acquisition*, pages 183–210.

Han, C.-H. and Hedberg, N. (2008). Syntax and semantics of it-clefts: a tree adjoining grammar analysis. *Journal of Semantics*, 25:345–380.

Headden, III, W. P., Johnson, M., and McClosky, D. (2009). Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Horst, J. S., McMurray, B., and Samuelson, L. K. (2006). Online processing is essential for learning: Understanding fast mapping and word learning in a dynamic connectionist architecture. In *Proceedings of the 28th annual meeting of the Cognitive Science Society*.

Huttenlocher, J., Haight, W., Bryk, A., Seltzer, M., and Lyons, T. (1991). Early vocabulary growth: Relation to language input and gender. *Developmental Psychology*, 27(2):236–248.

Johnson, M., Demuth, K., and Frank, M. (2012). Exploiting social information in grounded language learning via grammatical reduction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 883–891, Jeju Island, Korea. Association for Computational Linguistics.

Johnson, M., Demuth, K., Frank, M., and Jones, B. (2010). Synergies in learning words and their referents. In Lafferty, J., Williams, C. K. I., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 1018–1026.

Jones, B., Andreas, J., Bauer, D., Hermann, K.-M., and Knight, K. (2012a). Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of the 24th International Conference on Computational Linguistics*.

Jones, B., Johnson, M., and Goldwater, S. (2012b). Semantic parsing with Bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 488–496. Association for Computational Linguistics.

Jones, B. K., Johnson, M., and Frank, M. C. (2010). Learning words and their meanings from unsegmented child-directed speech. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 501–509.

Kate, R. J. and Mooney, R. J. (2006). Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 913–920.

Kim, J. and Mooney, R. J. (2010). Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 543–551.

Kim, J. and Mooney, R. J. (2012). Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL '12)*, pages 433–444.

Kim, J. and Mooney, R. J. (2013). Adapting discriminative reranking to grounded language learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-2013)*, pages 218–227.

Knight, K. and Graehl, J. (1998). Machine transliteration. *Computational Linguistics*, 24(4).

Knuth, D. E. (2005). Generating all tuples and permutations. *The Art of Computer Programming, Volume 4, Fascicle 2: Generating All Tuples and Permutations*.

Kollar, T., Tellex, S., Roy, D., and Roy, N. (2010). Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction*, pages 259–266.

Krishnamurthy, J. and Kollar, T. (2013). Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association of Computational Linguistics*, 1:193–206.

Krishnamurthy, J. and Mitchell, T. M. (2012). Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765.

Kurihara, K. and Sato, T. (2006). Variational Bayesian grammar induction for natural language. In *Proceedings of the 8th International Colloquium on Grammatical Inference*.

Kwiatkowski, T., Choi, E., Artzi, Y., and Zettlemoyer, L. (2013). Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556.

Kwiatkowski, T., Goldwater, S., Zettlemoyer, L., and Steedman, M. (2012). A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Landau, B., Smith, L., and Jones, S. (1998). Object shape, object function, and object name. *Journal of Memory and Language*, 38(1):1–27.

Landau, B., Smith, L. B., and Jones, S. S. (1988). The importance of shape in early lexical learning. *Cognitive Development*, 5:287–312.

Lany, J. and Saffran, J. R. (2010). From statistics to meaning: Infants' acquisition of lexical categories. *Psychological Science*, 21(2):284–291.

Lautemann, C. (1988). Decomposition trees: Structured graph representation and efficient algorithms. In Dauchet, M. and Nivat, M., editors, *CAAP '88*, volume 299, pages 28–39. Springer Berlin Heidelberg.

Le, P. and Zuidema, W. (2012). Learning ccompositional semantics for open domain semantic parsing. In *Proceedings of the 24th Intenational Conference on Computational Linguistics (COLING)*.

Lepore, E. and Ludwig, K. (2007). *Donald Davidson's truth-theoretic semantics*. Clarendon Press, Oxford, UK.

Li, P., Farkas, I., and MacWhinney, B. (2004). Early lexical development in a self-organizing neural network. *Neural Networks*, 17:1345–1362.

Li, P., Zhao, X., and MacWhinney, B. (2007). Dynamic self-organization and early lexical development in children. *Cognitive Science*, 31:581–612.

Liang, P., Jordan, M. I., and Klein, D. (2009). Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, pages 91–99.

Liang, P., Jordan, M. I., and Klein, D. (2011). Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*.

Littschwager, J. C. and Markman, E. M. (1994). Sixteen- and 24-month-olds' use of mutual exclusivity as a default assumption in second-label learning. *Developmental Psychology*, 30(6):955–968.

Lu, W., Ng, H. T., Lee, W. S., and Zettlemoyer, L. S. (2008). A generative model for parsing natural language to meaning representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Luong, M.-T., Frank, M. C., and Johnson, M. (2013). Parsing entire discourses as very long strings: Capturing topic continuity in grounded language learning. *Transactions of the Association for Computational Linguistics*, 1(3):315–323.

MacWhinney, B. (1989). Competition and lexical categorization. *Linguistic Categorization: Current Issues in Linguistic Theory*, 8:195–242.

MacWhinney, B. (2015). Lawrence Erlbaum, Mahwah, New Jersey, 3 edition.

Markman, E. and Wachtel, G. (1988). Children's use of mutual exclusivity to constrain the meanings of words. *Cognitive Psychology*, 20:121–157.

Markman, E. M. (1989). *Categorization and naming in children*. MIT Press, Cambridge, MA.

Martin, S. and White, M. (2011). Creating disjunctive logical forms from aligned sentences for grammar-based paraphrase generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation (MTTG)*, pages 74–83.

Matuszek, C., Fitzgerald, N., Zettlemoyer, L., Bo, L., and Fox, D. (2012). A joint model of language and perception for grounded attribute learning. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1671–1678.

Maurits, L., Perfors, A. F., and Navarro, D. J. (2009). Joint acquisition of word order and word reference. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*.

Mayer, M. (1969). *Frog, where are you?* Dial Press, New York.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530.

Merriman, W. E. (1999). Competition, attention, and young children's lexical parsing. *The emergence of language*, pages 331–358.

Mervis, C. (1987). Child-basic object categories and early lexical development. *Concepts and conceptual development: Ecological and intellectual factors in categorization*, pages 201–233.

Minka, T. (2000). Estimating a Dirichlet distribution. Technical report, M.I.T.

Naigles, L. R. (1990). Children use syntax to learn verb meaning. *Journal of Child Language*, 17:357–374.

Nesson, R. and Shieber, S. (2006). Simpler TAG semantics through synchronization. In *Proceedings of the 11th Conference on Formal Grammar*, pages 129–142.

Nesson, R., Shieber, S., and Rush, A. (2006). Induction of probabilistic synchronous tree-insertion grammars for machine translation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 128–137.

Niyogi, S. (2002). Bayesian learning at the syntax-semantics interface. In *Proceedings of the 24th Annual Conference of the Cognitive Science Society*.

Pavlidis, T. (1972). Linear and context-free graph grammars. *Journal of the Association for Computing Machinery*, 19(1):11–23.

Pinker, S. (1989). *Learnability and cognition*. The MIT Press, Cambridge, MA.

Plunkett, K., Sinha, C., Moller, M. F., and Strandsby, O. (1992). Symbol grounding or the emergence of symbols? Vocabulary growth in children and a connectionist net. *Connection Science*, 4:293–312.

Poon, H. (2013). Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 933–943.

Quine, W. (1960). *Word and object*. MIT Press., Cambridge, MA.

Reddy, S., Lapata, M., and Steedman, M. (2014). Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics (TACL)*.

Regier, T. (2005). The emergence of words: Attentional learning in form and meaning. *Cognitive Science*, 29:819–865.

Rosch, E., Mervis, C. B., Gray, W. D., Johnson, D. M., and Boyes-Braem, P. (1976). Basic objects in natural categories. *Cognitive Psychology*, 8:382–439.

Rounds, W. (1970). Mappings and grammars on trees. *Mathematical Systems Theory 4*, pages 257–287.

Roy, D. K. and Pentland, A. P. (2004). Learning words from sights and sounds: A computational model. *Cognitive Science*, 26(1):113–146.

Schafer, G. and Mareschal, D. (2001). Modeling infant speech sound discrimination using simple, associative networks. *Infancy*, 2(1):7–28.

Shieber, S. M. (1984). Direct parsing of ID/LP grammars. *Linguistics and Philosophy*, 7(2):135–154.

Shieber, S. M. (2004). Synchronous grammars as tree transducers. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+ 7)*.

Shieber, S. M. (2014). Bimorphisms and synchronous grammars. *Journal of Language Modelling*, 2(1):51–104.

Siskind, J. M. (1996). A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61(1-2):39–91.

Smith, K., Smith, A. D. M., and Blythe, R. A. (2011). Cross-situational learning: An experimental study of word-learning mechanisms. *Cognitive Science*, 35(3):480–498.

Smith, L. B. (2000a). Avoiding associations when it's behaviorism you really hate. *Becoming a word learner: A debate on lexical acquisition*.

Smith, L. B. (2000b). How to learn words: An associative crane. *Breaking the word learning barrier*, pages 51–80.

Smith, L. B., Jones, S. S., Landau, B., Gershkoff-Stowe, L., and Samuelson, L. (2002). Object name learning provides on-the-job training for attention. *Psychological Science*, 13(1):13–19.

Sowa, J. F. (1976). Conceptual graphs for a data base interface. *IBM Journal of Research and Development*, 20(4):336–357.

Steedman, M. (2000). *The syntactic process*. MIT Press, Cambridge, MA.

Stromqvist, S. and Verhoven, L. (2004). *Relating events in narrative volume 2: Typological and contextual perspectives*. Lawrence Erlbaum Associates, Mahwah, NJ.

Tang, L. R. and Mooney, R. J. (2000). Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora(EMNLP/VLC-2000)*, pages 133–141.

Tardif, T., Shatz, M., and Naigles, L. (1997). Caregiver speech and children's use of nouns versus verbs: A comparison of English, Italian, and Mandarin. *Journal of Child Language*, 24:535–565.

Titov, I., Henderson, J., Merlo, P., and Musillo, G. (2009). Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of the 21st international joint conference on Artifical intelligence (IJCAI)*, pages 1562–1567.

Tomasello, M. (2001). Perceiving intentions and learning words in the second year of life. *Language acquisition and conceptual development*, pages 132–158.

Vogel, A. and Jurafsky, D. (2010). Learning to follow navigational directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 806–814.

Wong, Y. W. (2007). *Learning for semantic parsing and natural language generation using statistical machine translation techniques*. PhD thesis, University of Texas at Austin, Austin, TX.

Wong, Y. W. and Mooney, R. J. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting*, pages 439–446, New York City, NY.

Wong, Y. W. and Mooney, R. J. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-2007)*.

Xu, F. and Tenenbaum, J. B. (2007). Word learning as bayesian inference. *Psychological Review*, 114(2).

Yamada, K. and Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of the annual meeting of the Association for Computational Linguistics*, pages 523–530.

Yu, C. (2005). The emergence of links between lexical acquisition and object categorization: A computational study. *Connection Science*, 17:3–4.

Yu, C. (2006). Learning syntax-semantics mappings to bootstrap word learning. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*.

Yu, C. and Ballard, D. H. (2007). A unified model of early word learning: Integrating statistical and social cues. *Neurocomputing*, 70(13-15):2149–2165.

Yu, C. and Smith, L. B. (2007). Rapid word learning under uncertainty via cross-situational statistics. *Psychological Science*, 18:414–420.

Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*.