A Framework for Representing and Jointly Reasoning over Linguistic and Non-linguistic Knowledge

by

Arthi Murugesan

A Thesis Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the degree of
DOCTOR OF PHILOSOPHY
Major Subject: Cognitive Science

Approved by the
Examining Committee:

_____
Nicholas L. Cassimatis, Thesis Adviser

_____
Paul Bello, Member

_____
Selmer Bringsjord, Member

_____
Mark Changizi, Member

_____
Ron Sun, Member

Rensselaer Polytechnic Institute
Troy, New York

October, 2009
(For Graduation December 2009)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

My graduate studies would not have been possible if not for unusual circumstances provided by extraordinary people.

I'm ever grateful to my parents for all they have done for me, not the least of which is believing in my abilities and sending me to a far off land for higher studies when I wasn't so sure myself. Thank you Amma and Appa.

I express my sincere thanks to Prof. Patrick Winston to whom I owe my AI foundation. Prof Winston's kindness to the “stranger who showed up at his office door”, wanting to learn more about AI has started my academic career.

This thesis as well as most of my research is the result of the hard work of Prof. Nick Cassimatis. His unique perspective on AI has inspired me, while his patience during trying times has helped me persevere through the entire PhD program. I have had the true joy of completely enjoying my work and taking pride in it.

I am also very grateful to the patience and guidance offered by my doctoral committee. Prof. Selemer Bringsjord has been very supportive of my research; his critical questions have led me to define my work more precisely. Dr. Paul Bello has been both a friend and an advisor. Prof. Ron Sun has been patient through the evolution of this thesis. Prof. Mark Changizi has always been ready to listen and to point out further directions to carry forward the research.

I would like to thank teachers at Rensselaer Polytechnic Institute who have helped fine tune my research. I gained immensely from Dr. Bram van Heuveln’s logic course. I’m grateful to have taken Prof. Jim Hendler’s semantic web course, which helped me view my research in a bigger context.

I would like to acknowledge my lab partners, Magda Bugajska, Scott Dugas, Perrin Bignoli, Miles Shuman and Unmesh Kurup, for the interesting Polyscheme lab-meeting discussions, which have sparked several sections of this research. Also, the friendly camaraderie of the students of Cognitive Science department was always there to provide a relaxed background during the difficult times of my research. I would like to specially thank Prof. Wayne Gray, who made sure that I was comfortable in the graduate program.

The staff of the Cognitive Science department have been supportive and in large part been responsible for the smooth functioning of my graduate school paperwork. I would like to acknowledge the outstanding work done by Betty Osganian, Paula Monahan and Cheryl Keefe.

Finally, I would like to thank the two people who have kept me sane and happy with their love, my baby sister, Akila and my husband, Bharathi Krishna.

ABSTRACT

Natural language poses several challenges to developing computational systems for modeling it. Natural language is not a precise problem but is rather ridden with a number of uncertainties in the form of either alternate words or interpretations. Furthermore, natural language is a generative system where the problem size is potentially infinite. This combination of uncertainty and generativity challenges existing computational solutions.

In order to model natural language, innovation is required at the fundamental level of computational algorithms. Existing algorithms are designed either to handle uncertainty or to handle generativity, not both. For example, Bayesian networks treat uncertainty in a generic mathematically well-founded fashion, but are however restricted to a fixed size and configuration during inference. On the other hand, conventional PCFG parsers like Earley parsers are not restricted to a particular finite set of possible words, but are incapable of handling other uncertainties as those brought up by the semantics of sentences.

Our approach to providing a computational framework capable of modeling natural language understanding is two-fold. Our first aim is to propose a formalism that is capable of representing various kinds of uncertainties, for example word sense ambiguity in syntax and uncertainty in referent resolution in semantics, in the same generic fashion. To follow up, we propose a working inference algorithm that overcomes the problems with large (often infinite) problem spaces entailed by such mechanisms.

Specifically, this thesis proposes a formalism, generative satisfiability language (GenSAT), capable of representing constraints over potentially infinite domains. An inference algorithm, GenDPLL, is then defined which is guaranteed for a well-defined sub-class (increasing cost theory, relevant models) of the possible problems expressed in GenSAT. In order to show that GenSAT and GenDPLL are capable of representing natural language, as a proposed first step, this thesis demonstrates how Probabilistic Context Free Grammar (PCFG) can be represented in GenSAT.

However, GenSAT is a fairly low-level representation, making the representation of PCFG constraints directly in GenSAT difficult to comprehend. Therefore, we have

defined an intermediate language, generative probabilistic language (GenProb) that abstracts over the details of GenSAT in a level of abstraction close to the conditional table representation in Bayes nets. GenProb is interesting in its own right, because it is well-defined with intuitive semantics and helps show case several properties of the problem encoded, like the property of increasing cost. The thesis then elaborately details the translation of a problem encoded in GenProb theory to a corresponding problem in the GenSAT language. The final chapter describes in detail how a PCFG problem can be represented in GenProb.

Hence by representing a linguistic grammar in the proposed formalism, GenProb which can be translated to GenSAT, and defining an inference algorithm GenDPLL for this language, we are able to show that language can be parsed using algorithms that unify syntax and semantics.

1. Introduction

Humans are able to combine information from ostensibly disparate sources in order to better comprehend the world around them. For example, the sense information from the eyes and the ears are combined at every instant - tasks such as road crossing would be far more difficult without such integration. Language understanding is also one such integrated task, where context and background information have been shown to interact with linguistic constraints (Langacker 1999; Jackendoff 2002; Pinker and Jackendoff 2005). For example, in order for a system to answer a question like "What is the weather going to be for the next Red Sox game?" information beyond linguistic constraints and the parse tree of this sentence is required. The knowledge of weather being associated with a location in this case the location of the game, and more specific information of that particular game like being played by Red Sox and occurring as the next game among a series of games relative to the current time, are all essential in understanding the question, let alone answering it.

Background knowledge and context are not the only sources of information required in sentence understanding. Some of the other prominent sources shown include vision and social cues (Frazier and Rayner 1987; Hall and McKevitt 1995). For example a sentences like "I love that dress", does not carry a lot of meaning by itself. Visual information of the dress like color, cut, fall, texture, and the listener's aesthetics help assign the sense of fashion of the speaker. Also depending on the social context, the speaker might be complimenting the person wearing the dress, or in the setting of a shopping spree be declaring one's intention of buying it.

1.1 Uncertainty in Language

One reason for why language understanding is difficult is that several forms of uncertainties are prevalent in language. For example, words spoken at a noisy bar are difficult to comprehend. Even hand-written words are at times difficult to recognize. Given we recognize the word; it could still mean one of its few senses. For example

the word “can” could either be a noun, a verb or a modal operator, and even when restricted to one part of speech like noun, “can” has the finer distinctions of being either an airtight sealed container or a plumbing fixture in a lavatory. Example usages of the word “can” is shown in figure 1.1.

```
A can of paint costs ten dollars.
She had to can the tomatoes over the weekend.
The author can present a convincing argument.
```

Figure 1:1 Example usages of "can"

Among the ambiguities in language, syntactic ambiguity is more difficult because of the generative nature of the problem. Also, syntactic disambiguation is at the intersection of syntax and semantics leading to a number of interesting results.

1.1.1 Syntactic Ambiguity

Given the grammar and lexical components of a sentence, the parsing process has the capability of producing more than one parse tree, among which one of them is considered the correct interpretation. Syntactic ambiguity is this uncertainty in choosing the correct tree structure among the several possibilities.

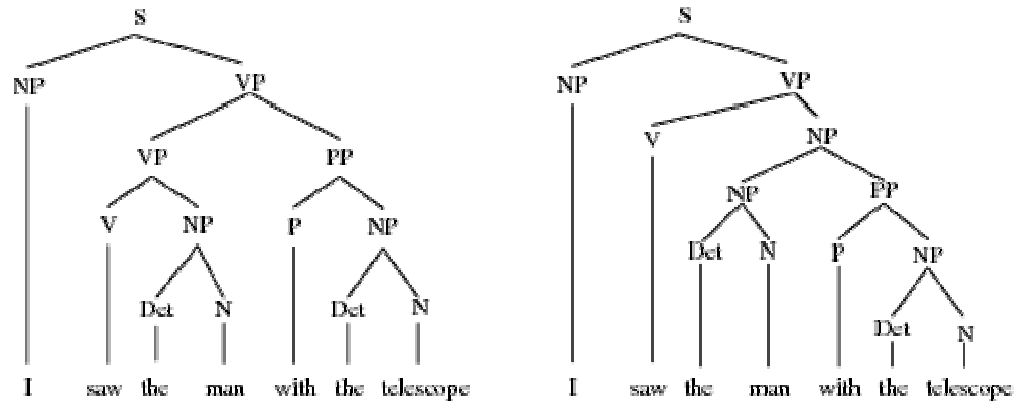

Figure 1:2 Two different interpretations of the same sentence “I saw the man with the telescope”

The two tree structures shown in figure 1.2 correspond to different interpretations of the sentence “I saw the man with the telescope”(Hindle and Rooth

1993). The first parse tree refers to the speaker seeing the man through or with the aid of the instrument telescope, while the second parse tree refers to the speaker seeing the man who is carrying the telescope. Though both interpretations of the sentence are possible, by uttering this sentence the speaker has tried to convey only one of those concepts, leading to uncertainty of what the speaker meant.

1.2 Current Approaches to Overcoming Problems due to Uncertainty in Parsing

Several approaches have been tried to handle the problem of uncertainty in NLP. There are various statistical techniques available to solve individual components of language understanding like hidden Markov models (HMM) for speech recognition (Hirsch and Pearce 2000). However, the accuracy of these techniques level out at values which are significantly lower than human performance. Furthermore, customized metrics for the individual components of NLP have been developed, like precision and recall for parsing, without regard to how these metrics correlate to progress of the field of natural language understanding as a whole. Therefore, several of these metrics do not reflect any corresponding progress in NLP.

The general theme behind recent improvements in accuracy has been using more information to resolve the uncertainties. The new information includes new sources of information and information in formats other than statistical (logical for e.g.). This section presents an overview of existing methods for handling syntactic ambiguity along with their limitations and recent improvements.

1.2.1 Overcoming Syntactic Ambiguity

The current state of the art statistical parsers have improved in accuracy from about 75% to 90% after including lexicalized statistics (Charniak 1997), which is statistics on the relationship between the symbol expanded and one of its children chosen as the head. Though there has not been such a marked improvement in accuracy based on linguistic phenomena like favoring right branching and collecting statistics on

semantically marked complement adjunct phrase distinctions, Collins(2003) shows minor improvements in accuracy.

Syntactic parsing has appeared to reach a plateau of improvement. Though precision recall metrics have shown drastic improvements of about 93% (Charniak 1997; Charniak 2000), the percentage of sentences that are parsed completely correctly (called the correct match (CM)) is still at about 25 to 40%. TRIPS system (Allen, Schubert et al. 1994; Ferguson and Allen 1998; Allen, Ferguson et al. 2001) which uses domain specific knowledge to aid parsing has shown about 65% CM accuracy on sentences with length greater than or equal to 2 words but is only 37.5% accurate then the length of the sentence is above 7 words. (Klein and Manning 2003) have shown precision recall of about 90% but complete match accuracies of only 30%. (Montazeri, Ghassem-Sani et al. 2006) shows an example system that has complete match accuracy of about 40%. Furthermore (Klein and Manning 2003) shows an analysis of how improvements in current methods increase precision recall metrics at the expense of complete match accuracy. Hence the complete match accuracy of parsing has reached a plateau of about 40%.

1.2.2 The Common Theme of Integrating Various Types of Information

Even though there are clearly cases where information from several aspects of NLP (like semantics) influence and in some cases is required to solve a sub problem (like part of speech tagging), this interaction is hard to conceptualize and reason over within a purely statistical system. The accuracy of purely statistical systems reaching an apparent maximum plateau has been attributed to this lack of integration. The general trend of improvement in statistical systems as seen in last section has been caused by integrating non-statistical information (both from various sources and in varying formats) to help disambiguate more accurately.

1.3 Sources of Information for Disambiguation

The existence of uncertainties in every aspect of language understanding makes it computationally complex. Though these uncertainties exist, other sources of

information besides language provide several cues that help resolve these uncertainties. World knowledge and perceptual information about the immediate environment are examples of these sources of information used for disambiguation.

1.3.1 World Knowledge

World knowledge can help resolve uncertainties like syntactic ambiguity and referent resolution. For example, both the sentences in figure 1.3 following “Dave hid Paul’s keys” refer to different people by the same pronoun “He” (Cassimatis 2008; McShane 2008).

(i) Dave hid Paul’s keys. He often jokes with him.

(ii) Dave hid Paul’s keys. He was drunk.

Figure 1:3 Reference resolution of pronoun "he"

In the first case “He” refers to Dave because hiding keys can be a joke that Dave plays on Paul. However in the second case, since a drunken person should not drive and the keys of a car are needed for driving it, the most possible interpretation is that Paul was drunk and Dave hid Paul’s car keys. Therefore, the pronoun “He” in the second case refers to Paul and not Dave. In this example, knowledge about the intention of hiding the keys helps resolve the referent of the pronoun. World knowledge like dictionary and word net have also shown to improve accuracies (Yarowsky 1992; Josef and Robert 2002; Simone Paolo and Michael 2006).

1.3.1.1 Beliefs and Intention

Besides being able to disambiguate among possible referents and possible interpretations of sentences, humans are also able to allow errors in the beliefs of the speaker and still interpret the sentence correctly. Consider a scenario in which two people in a bar are talking about a person seated in the nearby table (Cassimatis 2008). Since the speaker is unfamiliar with the person they are conversing about, she points to him as the person with “gin”. The listener, though she knows the person is actually drinking “sprite” (from when she overheard him ordering say), infers that

the speaker is referring to this particular person because no one else at the table is drinking a clear liquid. In this scenario the listener also infers that the speaker believes the person to be drinking gin instead of sprite. Even though there is a discrepancy between that which is true in the world or rather that which the listener knows and what the speaker utters, the listener is able to reason about the beliefs of the speaker and understand a potentially conflicting or false statement. Common examples of language that require reasoning with another person's beliefs include detecting sarcasm, persuasion or deceit and understanding speech acts.

1.3.2 Perceptual Sensory Information

Language is often understood in association with sensory and perceptual information. The ability to refer to the concepts of pain, color and heat through words is an example of this association. There are several aspects of perception that are used for resolving uncertainties in language.

1.3.2.1 Auditory Information

Tone of voice including pitch and loudness provide us with valuable information about emphasis, intonation and the mood of the speaker. For example, "Did *she* do that?" emphasizes on the disbelief of the speaker that the person *she* could have performed that act. However if the sentence was stressed differently as in "Did she do *that*?" the person is not being questioned as much as the action itself is being questioned regardless of who the person is. Examples described (Selkirk 1984; Selkirk 1995) include the sentences shown in fig 1.4. Such clues of intonation and mood are required for uncertainty resolution at several levels including syntactic ambiguity and semantics of people's beliefs.

```
Mary bought a book ten about [BATS].
(What did Mary buy a book about?)
Mary bought a book [about BATS].
(What kind of book did Mary buy?)
Mary bought [a book about BATS].
```

```
(What did Mary buy?)
Mary [bought a book about BATS].
(What did Mary do?)
[Mary bought a book about BATS].
(What's been happening?)
```

Figure 1:4 Variations of meaning expressed by intonations

1.3.2.2 Visual Information

Visual information gives us insights into two fundamental aspects of the world required to understand a sentence: its speaker and the immediate environment. Lip reading, hand gestures, facial expressions and tracking where the speaker's eyes are pointing can help us resolve various uncertainties in language. Uncertainties in speech recognition can be minimized by using information from lip reading and incorporating expectations for objects present in the immediate environment.

Visual information about the immediate environment also helps disambiguate among possible interpretations of structure and referent resolution. For example, the sentence "I would like that burger" pointing to a particular entry on the menu card when ordering to a waiter, is not ambiguous because the type of burger can be obtained through visual information.

1.4 Difficulties with Further Integration

Our take on the field, is that current statistical approaches have two problems that make integration, as motivated by the previous section, difficult: 1. they have weak, inexpressive formalisms (e.g. PCFG Vs Lexicalized PCFG) and 2. richer formalisms make it more difficult to create inference algorithms with properties like scalability and guarantee on the results. There are several obstacles to increasing the power of the statistical information capturing formalisms and making inferences over them. These problems can be illustrated in the domain of parsing.

1.4.1 Inexpressive Formalism

Propositional formalisms which are the norm of efficient SAT solvers are not powerful enough to express the kinds of relations required by parsing. In propositional formalism, words and phrases are represented as propositions like NN_{dog}, NN_{the_dog}, NN_{the_man} and $VP_{walked\_in\_the\_park}$. Every proposition (usually denoted by p, q or r) is unique and cannot be compared with other propositions. Thus the intuition that certain phrases are more similar to each other is not captured, for example NN_{the_man} being more similar to NN_{the_dog} as compared to $VP_{walked\_in\_the\_park}$ is not captured in this formalism.

Furthermore, the grammar rules that capture regularities of phrases cannot leverage on the similarity of phrases, or the categories of phrases. Hence the rule that say any NP followed by a VP is a sentence i.e. $S \rightarrow NP\ VP$ has to be individually written like

$$S_{the\_man\_walked\_in\_the\_park} \rightarrow NN_{the\_man}\ VP_{walked\_in\_the\_park}$$

$$S_{the\_dog\_walked\_in\_the\_park} \rightarrow NN_{the\_dog}\ VP_{walked\_in\_the\_park}$$

This method of encoding requires a large number of rules, a number proportional to the size of the vocabulary, to capture the grammar. Also because this kind of generality is not captured, novel words are not handled by these rules. For example if a novel phrase say NN_{the_dax} occurs, “the dax walked in the park” will not be inferred, though intuitively this phrase is similar to the two sentences about the man and the dog walking in the park.

Relational formalisms on the other hand, encode generalizations and have the expressive power of capturing formalisms like context free grammars. However the inference algorithms over these relational formalisms generally do not scale well, particularly in domains were the number of objects (in the case of language the number of words and phrases) is large.

Many current systems typically handle inference over relational formalisms by requiring the declaration of objects of the domain in advance and converting the relational formalism to propositional formalism (Loveland 1968; Wolf 1998; Singla

and Domingos 2006). This approach has scaling issues as the number of objects that can satisfy generalizations captured by the relational formalism increases. For example, a simple rule that captures the possibility that an agent can move to the adjacent location during the next time slot is written as

$$Agent(?x) \wedge Location(?x, ?initialLocn, ?tStart)$$
$$\wedge Adjacent(?initialLocn, ?nextLocn) \wedge NextTime(?tStart, ?tNext)$$
$$\rightarrow \quad Location(?x, ?nextLocn, ?tNext)$$

Lets say that the number of agents in a scaled down scenario is about 10, located in cells of a 10 x 10 x 10 discretized space occurring over say 100 time steps. The propositional representation of this problem requires one million propositions, leading to explosion of the search space.

The domain of language understanding also calls for a large number of objects because the number of words and phrases is orders of magnitude more than the conventional size of the problems handled by existing relational inference systems. Language also allows unknown objects (novel words) and infinite recursion. Such sentences can not be completely propositionalized as required by the inference of these relational systems.

1.4.1.1 Unknown objects

Syntactic parsers handle words, relations between words like grammar rules and phrases that were not anticipated before. Encountering novel phrases i.e. novel combination of words and to a lesser extent novel words is an everyday activity. Formalisms like CFG that captures the syntactic grammar heavily rely on the inference of new objects. A CFG rule of the format $NP \rightarrow DT\ NN$ licenses the inference of a previously unknown NP object when a DT and NN are observed.

Only a few systems (Milch, Marthi et al. 2004; Milch, Marthi et al. 2007) allow the introduction of unknown objects within the relational formalism. These systems however face serious problems because their languages allow infinite models. Also the existing systems are limited in that they do not have well defined guaranteed inference mechanisms.

1.4.2 Inference Algorithm

The existing inference algorithms like SAT solvers and MCMC operate on grounded constraints and generally do not scale to relational constraints. Certain SAT solvers, like LazySAT (Singla and Domingos 2006), operate efficiently on relational constraints; however, they are based on additional assumptions, for example that all objects in the world have to be declared beforehand. Probabilistic representation that allows unknown objects, like BLOGs, is in the spirit of the required representation. However the methods that allow unknown objects are limited and to our knowledge a guaranteed inference algorithm is not available for these representations.

1.5 Our Approach

In the thesis we would like to focus on improving the accuracy of syntactic parsing by integration with semantics and world knowledge. The reason for choosing this aspect of integration is threefold. Firstly syntactic parsing has reached the upper-bound of about 40% complete match accuracy though precision and recall accuracy has been shown to be 90%, providing substantial room for improvement(Klein and Manning 2003). The 40% complete match accuracy is the percentage of sentences that are parsed correctly. Secondly, key insights for the process of integration is available as the relationship between linguistics and world knowledge have been well studied both within the fields of linguistics and psychology (Talmy 1975; Jackendoff 1983; Talmy 1985; Talmy 1988; Jackendoff 1997). Finally, this integration is the cutting edge field of study as the immediate pay-off would be systems that can provide better response to queries [relevant to technologies like semantic web and search engines].

Our approach to integration of syntactic parsing and world knowledge is two fold. Firstly both syntactic processing information and world knowledge have to be represented within the same formalism. Secondly an inference algorithm that provides results over these constraints within practical time and with some guarantee of soundness of the result is required.

1.5.1 Representing Syntactic Grammar and World Knowledge within the Same Formalism

Even though there is a considerable difference between formalisms used to represent world knowledge and syntactic parsing, there is evidence in humans that the two types of knowledge are more closely knit. One reason for the optimism of an integrating formalism is the research in various fields (Minsky 1974; Johnson-Laird 1983; Jackendoff 1990; Croft and Cruse 2004) that have shown dualities between language and constructions used to understand the world. Cassimatis (2004) proposes a formalism by which language can be represented in terms of object interactions in the world. However, encoding a full coverage grammar, in terms of object interactions and other constructions used to understand the world, might prove to be a challenge.

In order to provide a proof of concept, a concrete example of representing syntactic parsing formalism in terms of world knowledge representation is required. Hence we are driven to choose an existing grammar and formalize the same in terms of non-linguistic formalisms. For reasons described in the next section, we choose Probabilistic Context Free Grammar (PCFG).

1.5.1.1 Reasons for the Selection of PCFG

Our approach is to map Probabilistic Context Free Grammar (PCFG) onto the common representation, rather than a highly structured lexical grammar like HPSG. This choice of grammar is primarily due to the difficulty of implementing HPSG on a large scale. HPSG requires a unification parser that is driven by identity constraints, which are computationally complex and difficult to scale over. Although, HPSG with its support for semantics is the ideal candidate, as a first step, we would like to implement Probabilistic Context Free Grammar (PCFG). PCFG is one of the most widely used grammars with rules in the orders of thousands, which challenges the system to be scalable. Furthermore, a majority of the state of the art NLP systems and recent academic pursuits (Collins 1996; Carroll, Briscoe et al. 1998; Charniak 2000) of improving parsing accuracy use PCFG as the de facto standard. Although

this is beyond the scope of the thesis, we believe that this work of mapping PCFG can be extended in the future to richer grammars like HPSG.

1.5.2 Mapping of PCFG onto a Representation for Semantics

Our approach to the mapping of PCFG onto a language that represents both syntax and semantics can be elaborated as follows. We propose to identify the constraints implicitly enforced by the PCFG formalism. These constraints must then be encoded in a common language that is also capable of representing semantics. Hence, we propose the language GenProb, which is a relational probabilistic formalism capable of inferring the existence of new objects as required by PCFG. Finally, an inference algorithm over GenProb has to be defined. Since there is no existing inference algorithm capable of handling a probabilistic relational model with unknown objects, we propose to convert GenProb to a weighted SAT representation that allows unknown objects, and defining an inference algorithm over this SAT representation. Generative SAT (GenSAT) is the weighted SAT representation described in this chapter and Generative DPLL (GenDPLL) is the inference algorithm defined over GenSAT. Figure 1.5 illustrates this process of mapping PCFG onto a representation that also supports semantics.

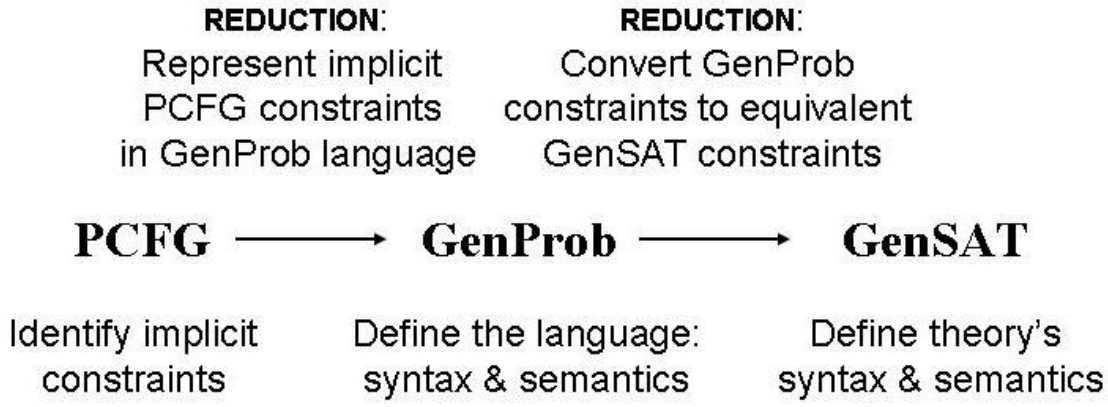

Figure 1:5 Mapping PCFG onto a representation that supports joint inference

1.5.2.1 Identifying Implicit Constraints of PCFG

Probabilistic Context Free Grammar is represented by rules, R, of the form:

$$\begin{aligned} X \rightarrow \quad & (prob_1)u_{11}u_{12} \dots u_{1m1} \\ & \mid (prob_2)u_{21}u_{22} \dots u_{2m2} \\ & \dots \\ & \mid (prob_n)u_{n1}u_{n2} \dots u_{nmn} \end{aligned}$$

A grammar G generates a language L(G) using the PCFG rules in R. There are several implicit constraints in the generation of a language. Our aim includes identifying and explicitly formalizing these constraints. The functionality of a parser P is also heavily dependent on these implicit constraints. For a given string (I), the parser determines whether and how the string can be generated from the grammar (G), (i.e.) it determines if (I) is a member of L(G).

An example of the implicit constraints enforced by the PCFG rules is the order of its arguments. This order of arguments determines the word order of a string that forms a valid sentence. Similarly all the other implicit constraints enforced by PCFG rules must be identified and be formalized explicitly.

1.5.2.2 Reduction of PCFG to Generative Probabilistic Formalism

Every implicit constraint identified in PCFG formalism has to be explicitly coded in language used to represent semantics. There are several key properties of PCFG that this language must be able to capture. PCFG encodes probabilistic relationships which must be supported by the language. Furthermore, PCFG allows the recursive expansion of non-terminals leading to possibly infinite models, hence the language must also be able to capture possibly infinite models. Therefore, we define a generative probabilistic language (GenProb) that has the characteristics of probabilistic and relational formalisms.

1.5.2.3 Defining the Common Language for Syntax and Semantics, GenProb

We define Generative Probabilistic language (GenProb), which is a probabilistic relational representation that captures features of both the relational first order representation and the causal Bayesian networks. Similar representations with the intent of combining probabilistic graphical models and logic have been proposed (Domingos and Richardson 2004; Richardson and Domingos 2006). However, these do not allow reasoning over unknown objects as required by PCFG. GenProb is designed to overcome these limitations.

1.5.2.4 Inference over GenProb through Translation to Generative SAT Constraints

Now that we have defined the GenProb language and proposed to encode the PCFG constraints in this language, the next step of our approach is to define an inference algorithm over GenProb. This research is motivated by exact inference algorithm that provides guaranteed results rather than approximations. Existing exact algorithms that infer over probabilistic graphical model formalisms convert these grounded probabilistic models to weighted SAT constraints and then perform exact inference over these constraints (Sang, Beame et al. 2005; Singla and Domingos 2006). However, as GenProb is a relational formalism, it can not be converted to SAT constraints. Hence we perform a similar reduction of GenProb onto a weighted SAT formalism that allows the licensing of new objects, GenSAT, and propose an exact inference algorithm over GenSAT.

1.5.2.5 Defining Generative SAT Theory

GenSAT, which is short for Generative SAT, is similar to MaxSAT (Singla and Domingos 2006); however it differs from weighted MaxSAT theory in that it allows the representation of relational clauses. The various components of GenSAT are causal and logical constraints in the form of conditionals and evidence in the form of literal assertions.

1.5.2.6 Inference over GenSAT

The foremost motivation for representing syntax and semantics in the same language is to achieve integration of the two by reasoning over this representation to resolve ambiguities in language. Therefore, defining reasoning mechanisms that are scalable and provide guaranteed results are crucial to the successful integration of syntax and semantics.

We propose an exact inference algorithm (GenDPLL) that reasons over the relational GenSAT constraints. We also propose several optimizations of the algorithm that can be used to provide approximate results when required.

1.5.3 Overview of the Thesis

The research in this thesis is motivated by the problems introduced in language due to uncertainty. Our key approach for handling uncertainty in language is integration. The next chapter describes the underlying guaranteed inference mechanism (GenDPLL) and the simplified SAT theory that this inference algorithm operates on, GenSAT. Chapter three proposes the syntax of GenProb and describes the duality between a GenProb model and its corresponding GenSAT constraints. Finally chapter four of the thesis shows the application of the common language (GenProb) by encoding PCFG constraints in GenProb language and showing how inference over the GenProb constraints provides the same results as a PCFG parser.

2. Inference for Generative SAT theories

As described in the last chapter, our overall approach involves providing a language and inference mechanism for probabilistic inference that are powerful enough to jointly represent and reason over linguistic and non-linguistic constraints. To enable this inference, we will translate theories in this language to a weighted satisfiability problem and provide an algorithm for solving these problems. Because language involves unknown objects and thus potentially infinite models, we must overcome several difficult technical challenges to achieve this. In this chapter, we describe an approach to weighted satisfiability reasoning that addresses this problem. In subsequent chapters we will describe how to use this framework to conduct the probabilistic inference we require for our work on language.

As described earlier the common language used for the integration of syntactic and semantic knowledge should be relational and allow the representation of uncertainty. A number of important problems with uncertainty can be characterized as weighted relational satisfiability problems. Propositionalizing these weighted relational theories and making inference over them have been shown to be effective in several cases. However, in order to propositionalize relational representations existing approaches require declaring in advance all possible grounding objects of variables. In a domain like natural language understanding, where novel words and novel combinations of words come up, requiring the system to declare all possible objects is not feasible. In this chapter, we describe a theory of relational representation with unknown objects that can be used by a model finding inference algorithm.

Many important tasks can be cast as weighted relational satisfiability (SAT) problems. Propositionalizing relational theories and making inferences with them using SAT algorithms has proven effective in many cases. However, these approaches require that all objects in a domain be known in advance. Many domains, from language understanding to machine vision involve reasoning about objects that are not known beforehand. Theories with unknown objects can require models with infinite objects in their domain and thus lead to propositionalized SAT problems that

existing algorithms cannot solve. To address these problems, we characterize a class of relational generative weighted satisfiability theories (GenSAT) over potentially infinite domains and propose an algorithm, GenDPLL, for finding models of these theories. We introduce the notion of a relevant model and an increasing cost theory to identify conditions under which the GenDPLL algorithm is complete, even when a theory has infinite models.

2.1 Problems due to Objects not Known before Inference

Propositionalizing first-order theories and making inferences using satisfiability algorithms has proven an effective means of solving problems in many domains (Domingos and Richard- son, 2006; Jackson, 2000; Singla and Domingos, 2006). These methods generally assume that all the objects in a domain are known in advance and begin by translating first-order clauses into propositional clauses. However, many problems involve objects that are not known before inference begins. For example, visual inference must deal with objects that are initially unknown because they are occluded and parsing a sentence using context-free grammars involves reasoning about constraints on (potentially infinitely many) phrases that were not known in advance of parsing. Unknown objects pose several difficulties for using satisfiability methods to reason over relational theories.

2.1.1 Infinite Models

Theories of infinite size that express relations over unknown objects often require infinite models. For example, the formula,

$$Mammal(a) \wedge \forall x(Mammal(x) \rightarrow Mammal(mother(x))$$

(together with formulae stating that a mammal cannot be its own ancestor) require an infinite model because a's mother must also have a mother who must also have a mother, ad infinitum. Likewise, some context-free grammars with infinite numbers of rules and terminals can generate an infinite number of sentences. Since an algorithm cannot enumerate an infinite model infinite time, we must find a way of infinitely characterizing solutions to problems that have infinite models.

Furthermore, even if all models of a theory can be infinitely characterized; there may nevertheless be infinitely many such models. Complete satisfiability algorithms (e.g., those based on DPLL) over finite domains are guaranteed to halt because they perform exhaustive search through the space of possible models. Thus, developing model finding algorithms when there are infinitely many possible models poses additional difficulties over standard complete satisfiability algorithms.

Some approaches (Singla and Domingos, 2006) deal with the problem of large propositional translations by lazily grounding first-order clauses. However, their algorithm is not complete and their approach requires that all objects are known in advance. They thus do not deal with problems that have infinitely large and/or infinitely many models. Several approaches to probabilistic inference involve unknown objects and infinite domains. Infinite Markov Logic Networks (Domingos and Richardson, 2007) have been extended to infinite domains, but, to our knowledge, an algorithm has not been proposed for inference over these networks. Several other approaches (Kersting and De Raedt, 2001; Milch, Marthi, Sontag, Russell, and Ong, 2005; Muggleton, 1996) do not enable exact inference and/or involve graphical models that impose stronger restrictions on these networks (e.g. regarding the existence of directed cycles) than are imposed by satisfiability theories.

2.1.2 Overriding Existing Grounded Constraints

Another problem regarding the licensing of novel objects is that some constraints involving grounded objects may have to be modified to accommodate the new objects posited during inference. An example of a key concept that requires the modification of constraints is a phenomenon in the interpretation of causal constraints that we label *mandatory causation*. Mandatory causation constraint is a rule that states that an event cannot occur without at least one of its causes being true.

For example, the grass cannot be wet unless one of the causes of wetness is true: i.e. either it must have rained or a sprinkler near it must have been on. Hence the mandatory causation rule for the grass being wet is:

$$NOT\ SprinklerOn_1\ \wedge NOT\ Rain \rightarrow NOT\ GrassWet$$

However, novel objects introduced to the inference system can lead to a new possible cause, therefore requiring the modification of the mandatory causation rule. For example, if later on we come to realize that another sprinkler could have watered the grass, then the mandatory causation rule now becomes

$$NOT\ SprinklerOn_1\ \wedge NOT\ SprinklerOn_2 \wedge NOT\ Rain \rightarrow NOT\ GrassWet$$

This modification of constraints can be recast as the removal of certain constraints, in this case $NOT\ SprinklerOn_1\ \wedge NOT\ Rain \rightarrow NOT\ GrassWet$, and the addition of newer constraints, $NOT\ SprinklerOn_1\ \wedge NOT\ SprinklerOn_2 \wedge NOT\ Rain \rightarrow NOT\ GrassWet$, to the inference system. The effect of adding constraints to the system have been studied in the existing methods. However, removal of constraints can lead to previously rejected models becoming optimal (by lowering the cost of these models) and thus may render the existing answer models sub optimal.

This chapter describes an approach that addresses issues raised by unknown objects and enables in many cases inference over problems with infinite models. First, we propose the GenSAT language for expressing relational, weighted constraints over unknown objects. Even in many cases where GenSAT theories have infinite models, it is possible to identify infinite partial models of interest to specific inference problems. We introduce the notion of a relevant model of a GenSAT theory to define such models. Next, we describe GenDPLL, an algorithm for finding models of GenSAT theories. GenDPLL is a DPLL-like branch-and-bound algorithm that lazily posits new objects and instantiates clauses involving them. Finally, we prove that GenDPLL is guaranteed to find infinite relevant models of certain classes of GenSAT theories with infinite models, which we call increasing cost models.

2.2 Generative Satisfiability Theories

We propose a language for generative satisfiability (GenSAT) theories that express constraints over relations among objects and licenses the "generation" of new objects during inference. The following simple example illustrates the main aspects of the

language. It involves clauses that state that a telephone (whose ringer is on) receiving a call and the striking of a bell both lead to a ringing sound and that a phone ringer is only on when the phone's battery is not empty.

$$BellStrike\ (?x.\ ?t) \rightarrow (10)\ RingSound\ (?t),\ ?x$$

$$PhoneGetsCall\ (?p,\ ?t)\ \wedge\ RingerOn\ (?p,\ ?t)\ \rightarrow (7)\ RingSound\ (?t),\ ?p$$

$$RingerOn\ (?p,\ ?t) \rightarrow \neg\ EmptyBattery\ (?p,\ ?t)$$

The numbers "10" and "7" are the weights on the constraints represented by the clauses. Terms beginning with '?' are variables. Variables listed after the right-most literal are called posited variables and indicate the potential of an object to be generated. For example, ?x in the second clause indicates that if a ringing sound occurs, then a bell (unknown prior to inference) being stuck may have caused it. This will be made more precise below. Non-posited variables are implicitly quantified.

It is often the case that if an event occurs or a relation obtains, then it must have been caused. For example, in planning many problems, everything not true in the initial state must have been made true by the execution of an operator. This is often encoded with what we here call a mandatory support constraint stating that either one of the possible causes of an effect holds or the effect does not hold:

$$cause_1 \vee \ldots \vee cause_n\ \vee \neg effect$$

When unknown objects are potentially relevant, it is often not possible to create such a clause in advance because all possible causes are not known. For example, if the number of bells is not known in advance of reasoning, then the number of possible causes of a ring cannot be known. It is thus impossible to explicitly state these mandatory support constraints in many GenSAT theories. We informally describe how a GenSAT theory deals with these constraints and in the next section provide more detail.

In GenSAT, the possible supports of a literal are indicated by clauses with numbers after the arrow. These are called causal clauses. The first two clauses above are causal. The weight associated with a causal clause, is the cost of a fully grounded instance of that constraint being violated. Thus, if two bells strike and do not cause ringing sounds, then a cost of 20 accrues. When a literal (e.g. *RingSound(?t)*) occurs

on the right hand side of a causal clause, it must be implied by at least one other causal clause. We call such literals causal literals. Non-causal literals do not require a support. For example, since no *EmptyBattery* literal occurs on the consequent of a causal clause, a ringer on does not cause the battery being empty to be false. Non-causal clauses, which we will call logical clauses, are always interpreted to be hard constraints with infinite cost. They are syntactically distinguished from causal clauses in not having an explicit weight.

If not all possible supports for a literal are encoded by a theory, then one can neutralize mandatory causation with a clause for an unknown support. For example, mandatory support for *RingSound* literals can be neutralized with the clause

$$UnknownCauseOfRinging(?t) \rightarrow (\infty)\, RingSound(?t)$$

This encodes the fact that ringing can have a support that is not listed among those enumerated in the theory.

More formally, a GenSAT theory is a set of causal and logical clauses. Causal clauses are of the form:

$$C_1 \wedge \cdots \wedge C_l \rightarrow (?\omega),\ E_1, \cdots, E_m, ?p_1, \cdots, ?p_n$$

where ω is a positive scalar or 1 and the *C*'s and *E*'s are each either literals or negations of literals. Literals are of the form: *P(a1, … , an)*, where each argument is a term. Terms that are not variables are called objects. Logical clauses are of the form:

$$A_1 \wedge \cdots \wedge A_l \rightarrow B_l \wedge \cdots \wedge B_m, ?p_1, \cdots, ?p_n$$

where the *A*'s and *B*'s are each either literals or negations of literals. Logical clauses can have zero literals in the antecedent and one in the consequent, in which case they can be called facts. The fact $\rightarrow P$ can be abbreviated as *P*. Clauses with no posited variables can be abbreviated by omitting the comma.

2.3 Translation to Propositional Form

In this section, we specify a translation from a GenSAT theory into a (potentially infinite) set of propositional clauses. The translation facilities defining models of

GenSAT theories and is important in characterizing a procedure we propose for making inferences over GenSAT theories.

The translation of a GenSAT theory is the union of: 1. A set of propositional clauses formed by grounding clauses in the theory and 2. A set of clauses that capture mandatory support constraints implied by the other clauses in the theory. We illustrate translation with the GenSAT theory with the following four clauses.

$$Hammer(?h) \wedge Bell(?b) \wedge Hit(?h, ?b) \rightarrow (56)Ring(?b), ?b$$

$$Hammer(h)$$

$$Bell(b)$$

$$Hit(h, b)$$

$$Hammer(?h) \wedge Bell(?b) \wedge Hit(?h, ?b) \rightarrow (56)Ring(?b), ?b$$

$$Hammer(h)$$

$$Bell(b)$$

$$Hit(h, b)$$

The translation of this theory grounds variables with objects occurring as arguments of literals in the theory and splits causal clauses into two clauses, one logical and one causal. Reasons for this split will be discussed below.

$$Hammer(h) \wedge Bell(b) \wedge Hit(h, b) \rightarrow (56)CauseRing(h, b)$$

$$CauseRing(h, b) \rightarrow Ring(b)$$

$$Hammer(b) \wedge Bell(h) \wedge Hit(b, h) \rightarrow (56)CauseRing(b, h)$$

$$CauseRing(b, h) \rightarrow Ring(h)$$

$$Hit(h, b)$$

$$Hammer(h)$$

$$Bell(b)$$

The translation also includes clauses representing the constraint that ringing events must be caused by a hitting event:

$$\neg CauseRing(h, b) \rightarrow \neg Ring(b)$$

$$\neg CauseRing(b, h) \rightarrow \neg Ring(h)$$

This relatively simple expression of mandatory support constraints is enabled by splitting or "bifurcating" ground GenSAT clauses with the *CauseRing* literals. This is the purpose of splitting causal literals in the translation.

To illustrate a theory over unknown objects, consider adding a clause to the effect that ringing bells generates sounds:

$$Ring(?b) \rightarrow GenerateSound(?b, ?s), ?s$$

Adding this clause to the GenSAT theory above would add the following two clauses to its translation:

$$Ring(b) \rightarrow GenerateSound(b, bSound)$$
$$Ring(h) \rightarrow GenerateSound(h, hSound)$$

These clauses include terms *bSound* and *hSound* which, informally, are the sound generated by the ringing of *b* and of *h*. The following theory illustrates the possibility of infinite translations.

$$Mammal(a)$$
$$Mammal(?m) \wedge GiveBirth(?m, ?b) \rightarrow Mammal(?b), ?m$$

Its translation includes infinitely many clauses:

$$Mammal(a)$$
$$Mammal(aMother) \wedge GiveBirth(aMother, a) \rightarrow Mammal(a)$$
$$Mammal(aMotherMother) \wedge GiveBirth(aMotherMother,aMother) \rightarrow Mammal(aMother)$$
$$\ldots$$

We now describe the translation more precisely in terms of a function that maps a GenSAT theory onto a set of grounded propositional clauses. The definition of the grounding of a GenSAT theory depends on a skolem function *s* that uniquely maps posited variables to objects not in *P*. The purpose of this function is to produce objects that are inferred during inference and that thus do not occur in *P*. For example, in the mammal theory above, for example, $s(a) = aMother$.

The translation of a GenSAT theory *P* is the union of groundings of clauses in *P* to propositional clauses and the set of clauses representing mandatory support constraints for the theory.

$$Translation(P, s) = Grounding(P, s) \cup MCClauses(Grounding(P, s))$$

A clause is in $Grounding(P, s)$ if it can be generated by successive steps of grounding.

$$Grounding(P, s) = OneStepGrounding(P, Objects(P), s) \cup$$
$$OneStepGrounding(P, Objects(OneStepGrounding(P, Objects(P), s)), s)$$

A clause is produced in a step of grounding if it can be produced by assigning to variables objects that are either generated by a skolem function or are already in the translation.

$$OneStepGrounding(P, objects, s) = U_c ClauseGrounding(c, objects, s)$$

The grounding of a clause with respect to a set of objects is the set of clauses that can be formed by all the possible assignments of those objects to variables in the clause. Where c_l is a logical clause, $l \rightarrow r, \cdots p_i \cdots$, with variables $v_1 \cdots v_n$, first appearing in the clause in that order:

$$ClauseGrounding(c_l, objects, s) = \{l_{a,s} \rightarrow r_{a,s} \mid a \in A\}$$

where $l_{a,s}$ and $r_{a,s}$ are the antecedent and consequent of c_l formed by substituting non-posited variables in c_l according to the assignment a and skolem function s applied to posited variables.

For example, the grounding of $A(?x) \rightarrow B(?y), ?x$, with respect to s and objects $\{a,b\}$ includes (only) the following two clauses:

$$A(s(a)) \rightarrow B(a)$$
$$A(s(b)) \rightarrow B(b)$$

The grounding of causal clause, $c_s = l_s \rightarrow (\omega) r_s, \ldots p_i \ldots$ is similar except it is "bifurcated", as was illustrated previously and is described presently.

$$ClauseGrounding(cs, objects, s) = ls,a \rightarrow (\omega) m,$$
$$m \rightarrow rs,a$$

where m, called an intermediate literal, occurs nowhere else in the translation and is unique to c_s and a.

For example, the grounding of $P(?x) \rightarrow (15)\ Q(?y), ?y$, with objects $\{a,b\}$ is the following set of four clauses.

$$P(a) \rightarrow (15)ma,$$

$$ma \rightarrow Q(s(a))$$
$$P(b) \rightarrow (15)mb$$
$$mb \rightarrow Q(s(b))g$$

As mentioned above, the purpose of bifurcating a causal clause using an intermediate literal, m is that it enables the creation of clauses that express mandatory support constraints. A mandatory support constraint for a literal states that if none of the possible "supports" of l is true, then l is false. Specifically:
MCClause(l) = $(\cdots \wedge \neg mi \wedge \cdots) \rightarrow \neg l$, where the mi are the intermediate literals of causal clauses in the translation that have l in the consequent.
MCClauses(S) = *lMCConstraint(l),* where l ranges over all the literals in S that occur in clauses ground from causal literals.

For example, if *Rain()*$\rightarrow$ *(*∞*)Wet(g)* and *Sprinkler(g)* $\rightarrow$ *(*∞*)Wet(g)* are the only two causal clauses in the grounding of P, then the following mandatory support clause appears in the translation:

$$\neg m_{rain} \wedge \neg m_{sprinkler} \rightarrow \neg Wet(g)$$

where m_{rain} and $m_{sprinkler}$ are the intermediate literals formed in the grounding of the clauses in *P*.

Note also that any MaxSAT theory can be expressed as a GenSAT theory. MaxSAT theories are a conjunction of disjunctions (often called clauses) of the form $(t_1 \vee \ldots \vee t_n, \omega)$, where each t_i is a possibly negated propositional literal and *w* is a weight. MaxSAT theories can be straightforwardly translated into GenSAT theories by translating clauses into the GenSAT clauses with equivalent truth conditions: $(\neg t_1 \ldots \wedge \ldots \neg t_{n-1} \wedge t_n, \omega)$. This equivalence between implication and disjunction can also be used to convert a translation of a GenSAT theory into a MaxSAT constraint that has possibly infinitely many clauses, some of which have infinite many disjunctions.

Since a translation can have an infinite number of clauses with the same literal in the consequent, there can be mandatory causation constraints that have an infinite number of disjuncts. All other clauses have only infinite numbers of disjuncts

because they are translations of GenSAT clauses, all of which have relational disjuncts.

2.4 Models

Propositional translations of GenSAT theories motivate a straightforward approach to defining models of them. In many cases where the translation is infinite, it is possible to identify infinite models of a theory that are suitable for the purposes of inference. We introduce the notion of relevance in order to characterize such models.

A model for a GenSAT theory is simply an assignment of truth values to the ground literals in its translation. The cost of a model is the sum of the weights on the clauses that are unsatisfied under this assignment. A clause, $\ldots l_i \ldots \rightarrow \ldots l_j \ldots$, is unsatisfied if all the l_i are true and any one of the l_j is false. A solution to a GenSAT theory is a model whose cost is infinite and not exceeded by the cost of another model for that theory. We say that an object is an object of a model if it occurs in a literal assigned by the model. Except for the possibility of involving infinite numbers of literals, GenSAT models are similar to models of ordinary MaxSAT theories.

Satisfiability algorithms return models of SAT theories. This creates a problem for making inferences using GenSAT theories because they can have infinite models. In many cases, however, it is possible to identify "relevant" parts of models that suit many inferential purposes. In this section, we introduce and more precisely characterize the notion of a relevant model.

As a simple illustration, consider the GenSAT theory with two clauses, *Pet(?x)* $\rightarrow$ *(10)Loved(?x)* and *Dog(?x)* $\rightarrow$ *Furry(?x)*, and one fact, *Dog(d)*. Intuitively, we can infer from one fact from this theory: that the dog, *d*, is furry. There are however four models for this theory. In each of these models, the *d* is furry, but the models vary according to whether *d* is a pet or if *d* is loved.

Table 2.1 All possible models of the theory

Pet(d)	*Loved(d)*	*Dog(d)*	*Furry(d)*	*Cost*
0	0	1	1	0

0	1	1	1	0
1	0	1	1	10
1	1	1	1	0

Given what is known, however, whether d is a pet or loved cannot be inferred and can in no way affect whether d is furry. Ignoring the *Pet* and *Loved* literals, we have the one relevant model shown in table 2.2.

Table 2.2 Relevant model

Dog(d)	*Furry(d)*	*Cost*
1	1	0

Definition 1 A model of GenSAT theory P is relevant if it can be derived by eliminating all irrelevant truth value assignments from a model for P. A literal ℓ and a truth value assignment (ℓ, *tv*) are relevant in a model if l is a ground literal in a clause in P or if ℓ is in a grounding of a clause, C, that has a relevant literal and l's truth value can affect whether C is satisfied.

As an example of the second condition above, if $Furry(d)$ is false in a model, then $Dog(d)$ is relevant in that model because if $Dog(d)$ is true then the clause is unsatisfied. Since neither $Pet(d)$ nor $Loved(d)$ occur in the GenSAT theory and because they occur in no clauses in the translation that have a relevant literal, they are not assigned in any relevant models.

A relevant solution to a GenSAT theory is a relevant model whose cost is infinite and not exceeded by another relevant model.

2.5 Finding models

An algorithm for finding models of GenSAT theories must address several challenges. First, in cases where a theory has infinitely large models, care must be taken not to endlessly explore models with infinite size. Second, even in cases where there are no infinite models, the size of the propositionalized translation can be very

large. Finally, in cases where all objects are not known during inference, all possible supports for a literal cannot be known in advance either. Thus, an algorithm can only guess at the mandatory support constraint before exploring models, and it must potentially alter that guess as search proceeds.

The GenDPLL algorithm successfully addresses these issues under certain conditions described below. It is based on DPLL "branch and bound" (Borchers and Furman, 1999) algorithms. These algorithms perform a depth-first search of the space of possible models by branching on one literal at a time, first exploring one truth value for that literal and then the other. Each time DPLL assigns a truth value to a literal, it performs an elaboration step, which sets variables to values implied by current assignments. "Branch and bound" DPLL algorithms avoid needless search by ceasing to explore a partial assignment when its predicted cost exceeds the best model found so far (or some threshold set a priori). The time complexity of GenDPLL is the same as that of DPLL algorithm, exponential and NP-complete.

In order to not endlessly explore infinite-cost models, GenDPLL begins searching (GenDPLLInner) with a cost threshold such that partial models are excluded if their cost exceeds it. If no such models are found, search begins again with the threshold doubled. This can lead to parts of the search space being explored more than once, but this can often be addressed in practice by estimating the cost of the best model and choosing a threshold significantly above that estimate.

The main procedural differences between GenDPLL and DPLL occur during the elaboration step. Elaboration involves three elements that address the three problems with infinite models we have just described. First, in order to conserve space, GenDPLL lazily grounds GenSAT into weighted DPLL constraints. Second, the elaboration step uses a variable selection and unit propagation scheme that prevents infinite models from being explored unnecessarily. Finally, at any given time, it explicitly assumes that all the possible supports for a literal are known. When a new possible support for a literal is inferred, GenDPLL backtracks to the stage where it assumed that all supports for that literal were known. Thus, at any given

time, the assumptions about all supports being known for a literal are encoded and can be retracted. These aspects of GenDPLL are explained in further detail presently.

Algorithm 1 $GenDPLL(P, t)$

Require: P is a GenSAT theory and $t \geq 0$

Choose an arbitrary skolem function s.
$initialClauses = OneStepGrounding(Objects(P), P, s)$
$q =$ the empty queue
$m = GenDPLLInner(P, t, q, , intialClauses, 0, nil)$
if $(m \neq fail)$ **then**
  return m
else
  return $GenDPLL(P, 2t)$
end if

Algorithm 2 $GenDpllInner(P, threshold, q, assignment, propClauses, depth, best)$

Require: P is a GenSAT theory, q is a queue of propositionalized literals, $t > 0$, $maxNewObjects \geq 1$, $assignment$ is a partial assignment from propositional literals, $propClauses$ is a set of propositionalized clauses, $d \geq 0$, $bestModel$ is a model of P or nil.

$elaborate(P, assignment, propClauses, depth)$
$weight \leftarrow$ total cost of unsatisfied clause in $propClauses$ under $assignment$.
if there is a hard contradiction or $weight > threshold$ **then**
  return $fail$
else if q is empty **then**
  return $assignment$
else
  $next \leftarrow q.next()$
  $newAssignment \leftarrow GenDPLLInner(P,\ threshold,\ q,$
    $assignment[next/true],\ propClauses,\ depth,\ bestModel)$
  if $(newAssignment = fail)$ **then**
    return $GenDPLLInner(P, threshold, q, assignment[next/false],$
      $propClauses, depth, bestModel)$
  else
    return $GenDPLLInner(P, threshold, q, assignment[next/false],$
      $propClauses, depth, newAssignment)$
  end if
end if

Algorithm 3 Procedure $Elaborate(P, assignment, propClauses, depth)$

Require: P is a GenSAT theory, $assignment$ is a partial assignment of literals to truth values, $propClauses$ is a set of propositional clauses and $depth \geq 0$

$maxNewObjects \leftarrow$ an arbitrary nonzero

purge q of literals added at depth levels greater than $depth$

$n \leftarrow 0$

$oldMCCLauses = mcCLauses(D)$

repeat

$propClauses \leftarrow D \cup$ all relevant literals in $OneStepGrounding(P, objects(PropClauses))$

$newLiterals \leftarrow$ literals in D not in $assignment$ or already in q

add $newLiterals$ to q

$n \leftarrow n+$ number of newly posited objects in $newLiterals$

until $n < maxNewObjects$

$newMCCLauses \leftarrow mcClauses(D, P) - oldMCCLauses$

add to q all relevant literals in $newMCClauses$ not in $assignment$ or not already in q

$propClauses = propClauses \cup newMCCLauses$

move to front of q all known supports assumption literals in $newMCCLauses$

GenDPLL takes as input a GenSAT theory and a maximum cost for models it considers. Its output is one of the models with the lowest cost. In describing GenDPLL we make use of the functions introduced in the section on propositional translation.

GenDPLL maintains several data structures. $groundConstraints$ is the list of propositional constraints that have been translated from first-order GenSAT clauses. "The queue", or q, contains propositional literals to be branched on. If a literal is put on the queue in a partial model that has been backtracked from, it is removed from the queue during backtracking. Specifically, literals on the queue are associated with the depth of search (d) when they were put on the queue. When backtracking from a level, all literals from that level are removed from the queue.

2.5.1 Lazy Instantiation

GenDPLL begins by finding all grounded GenSAT constraints and adds the literals in them to the queue. It then performs an elaboration step (described in more detail below) that grounds conditionals involving the current literal and adds new relevant (according to the definition in section (4)) literals to the queue. If the elaboration step

finds that the current assignment's cost exceeds the threshold, failure is returned and literals put on the queue at this depth of search are removed. If there are no more literals on the queue, this assignment and weight are returned and the threshold is lowered to the new weight. If there are more literals to explore, GenDPLL "branches" on the next literal in the queue.

2.5.2 Finite elaborations

Since grounding can add new objects, which can itself lead to further grounded clauses, it is a potentially infinite process. Consequently, GenDPLL only adds an infinite number of objects ($maxNewObjects$) during each elaboration step. This prevents elaboration steps that do not terminate. Since literals that might lead to more objects being posited remain on the queue, they will be elaborated eventually, so long as this partial assignment is not backtracked from. The next section discusses conditions under which GenDPLL can be guaranteed to halt.

2.5.3 Mandatory causation

During the elaboration step, if a literal, l, is created for the first time and it appears on the right hand side of a ground causal rule, GenDPLL "assumes" all supports are known for it and creates a mandatory support constraint. Specifically, for a $P(a_1 \ldots a_n)$ the literal $AllSupportsKnownP_m(a_1 \ldots a_n)$(where m is the number of known-support literals that have been instantiated for $P(a_1 \ldots a_n)$) is added to the current assignment as true and the following constraint is added:

$$AllSupportsKnownP_m(a_1 \ldots a_n) \wedge NOT\ C_1 \wedge \ldots \wedge NOT\ C_m \rightarrow P(a_1 \ldots a_n)$$

Literals whose predicates begin with $AllSupportsKnown$. When another causal clause is added with $P(a_1 \ldots a_n)$ on the right hand side, the point in search where its known-supports assumption was made is backtracked to and the assumption is constrained to be false. Since known support assumptions should be inferred to be true or false only when new possible supports are added, unit propagations assigning them truth values are not permitted.

Thus, GenDPLL explicitly assumes that all the potential supports of a literal have been grounded and, when a new possible support is discovered, backtracks to the point where it makes this assumption.

2.6 Increasing Cost Theories

There are many cases where infinite models or models of infinite size or possible, but where an infinite model is guaranteed to be the best solution. For example, some context-free grammars generate infinite numbers of trees. However, if the parses are weighted such that each phrase in the parse adds a "cost" to that parse tree, then if there are any infinite parses at all, the parse with the best cost will be infinite. As another example, consider path search problems with an unbounded numbers of moves being possible. If there is a path of infinite length and there is a cost associated with each move, then the best (i.e., least costly) path will be infinite, even if there are paths of infinite length.

In cases such as these, where the cost of a theory's model grows as the size of the model grows, GenDPLL can find models of GenSAT theories even when there are infinitely many models and/or some have infinite size. In this section we will precisely characterize such "increasing cost theories" and shoe that GenDPLL is guaranteed to find infinite models of them, when they exist.

First, we show that GenDPLL is sound.

Proposition 1 GenDPLL only returns models whose assigned literals are all relevant.

Proof. When finding a model for a theory P, GenDPLL will only put on the following two kinds of literals on the queue: ground literals in clauses in P and relevant literals in conditionals ground from other literals on the queue that have been assigned. These are precisely the conditions of literal relevance. Since GenDPLL will only make assignments for terms in the queue, all literals in the assignments it forms will be relevant.

When models for a GenSAT theory increase in cost as objects are added, GenDPLL will halt. We can state this more precisely by introducing the notion of an increasing cost theory.

Definition 2 A GenSAT theory P is an increasing cost theory if for each object o that does not occur in P (i.e., for reach object that will be posited during inference), o is involved in at least once satisfied clause (which by definition will have a positive nonzero cost).

For example, the following clauses encode that people reach for food either because they are hungry or because their boss reached for food and failed and they desire to help their boss.

$$Hungry(?p) \rightarrow (100)ReachForFood(?p, ?f)$$

$$NOT\ Hungry(?p) \wedge ReachForFood(?boss, ?f)$$

$$\wedge\ FailedReach(?boss, ?f) \wedge WorkFor(?p, ?boss) \rightarrow$$

$$ReachForFood(?p, ?f), ?boss$$

The following clauses state the (weighted) constraint that a boss will not fail to reach for food.

$$NOT\ FALSE$$

$$FailedReach(?p, ?f) \rightarrow (100)FALSE$$

These clauses together with the fact $ReachForFood(a)$ yield models of infinite size. Specifically, a model is possible where a reaches for food because food was reached for by the boss of the boss of the boss (ad infinitum) of the boss of a reached for food because their boss reached for food because the boss of their boss reached for food. However, each posited boss failing to reach for food breaks the last clause above and thus adds to the cost of the model. As a consequence, an infinite model will have a lower cost. Since GenDPLL only posits infinitely many objects during each elaboration step, it will reject models when enough bosses were posited to either exceed the initial cost threshold or the cost of the best model so far.

Proposition 2 For increasing cost theories, GenDPLL terminates and when there is a relevant solution, GenDPLL returns one.

Proof. First, we show that for one of the relevant solutions, s, for an increasing-cost theory, GenDPLL will find it and then we show that it will do so in infinite time. If a literal, l, is relevant, it is, by definition, either because (1) it is a ground literal in P (in which case it is either put on the queue to be branched on or assigned at the

beginning of GenDPLL) or (2) it is relevant under an assignment of other ground literals in P or in partial assignments whose literals are already relevant. In the second case, the elaboration step either branches on or unit propagates values for these literals. These two cases correspond exactly to the definition of a relevant literal. Thus, unless a relevant solution has already been found, s will be found.

This will occur in infinite time because all models with higher cost will exceed the cost of either the threshold or the best model as objects and the cost of the unsatisfied constraints they participate in are added. Such constraints are guaranteed to exist by definition of an increasing-cost model.

It is interesting to note that weights on constraints generally make require infinite weighted-SAT solving to be slower than Boolean SAT solving because fewer models can be ruled out. In the case of theories with infinite models, however, weights are the key to halting in infinite time because for increasing cost theories, they lead to search being halted before every variable of an infinite model is assigned.

2.7 Conclusions

Many problems can be formulated using relational theories over known and unknown objects. Models of such theories must often involve infinite numbers of objects. Existing propositional satisfiability algorithms therefore cannot in general find models of theories over unknown objects because their propositional translations can involve infinite numbers of variables and hence have infinite numbers of models of infinite size. Further, some kinds of mandatory support constraints, which are important for many applications, cannot be explicitly stated because they involve potentially infinite numbers of variables.

We have proposed a language for Generative satisfiability (GenSAT) theories for dealing with these problems. GenSAT enables relational constraints to be formulated over potentially unknown objects. A translation of GenSAT theories into propositional constraints makes mandatory clause constraints implicit in GenSAT theories explicit.

We have introduced the notion of a relevant model that allows useful infinite models of some GenSAT theories with infinite propositional translations to be characterized. We have proposed the GenDPLL algorithm for finding such models. By lazily positing objects and instantiating constraints, by positing only infinite numbers of objects in each elaboration step and by making explicit assumptions about knowing all supports for an event, GenDPLL will find minimal-cost models for increasing cost theories, whose models are guaranteed to grow in cost as the number of objects they involve grows.

Table 2.3 Comparison of DPLL algorithms

	Format they inference over	Causal equivalent	Applications
DPLL	Propositional CNF formulae	Bayes nets with CPT entries of 1 or 0	Theorem provers
Weighted DPLL	Propositional CNF formulae with weights	Bayes nets	Probabilistic systems in which finite objects
GenDPLL	Relational formulae with weights	Bayes nodes that can have new nodes added to them	PCFG

Now that GenSAT theory and GenDPLL inference algorithm have been described, the next step of our approach to integrating syntax and semantics is to define the GenProb language, which is capable of representing both semantics and probabilistic context free grammar. After defining GenProb, we provide the rules that can translate a GenProb model to its corresponding GenSAT model. These steps are described in the next chapter.

3. GenProb

As mentioned earlier, this thesis aims to provide a framework for integrating syntactic, semantic and nonlinguistic information in natural language understanding. Our approach has been to develop a relational probabilistic inference language (which we call GenProb) for formulating constraints over linguistic and nonlinguistic information and to use a weighted SAT solver to make inferences over that language. In this chapter, we describe GenProb, which is a relational probabilistic representation that captures constraints both in the form of causal and logical relations. Some of the unique features of GenProb are its ability to allow the licensing of previously unknown objects and its ability to incorporate logical constraints within the probabilistic framework.

Combining first-order logical constraints and probabilistic graphical models has been shown to be an effective tool for compactly representing and performing inference in a number of domains (Domingos and Richardson 2004). For example, planning problems that involve a time component are best expressed in first-order logic formalism; however, the probability that a store on the way home holds a particular product (or a particular path is affected by traffic) is information best represented in a probabilistic graphical model. Combining these two representations (Berrada and Ferland 1996) has shown qualitative performance improvements in domains where both complex probabilistic information is required as well as where adaptable general constraints hold.

Combining first-order logic and probabilistic graphical models by propositionalizing relational theories has proven a promising approach for compactly representing and making inferences in many domains (Pedro Domingos & Richardson, 2006). Many of these approaches require all objects in the domain to be known in advance, although many important problems involve objects that are initially unknown. For example, probabilistic context-free grammars permit infinite derivations and thus all possible phrases cannot be known in advance. Objects tracked by machine vision systems can interact with occluded and thus initially unknown objects.

Theories over potentially unknown objects pose two problems for approaches based on propositionalization. Since all objects cannot be known in advance a theory cannot be fully propositionalized before inference. Second, languages that refer to unknown objects often entail models of infinite size. For example, function symbols with more than one argument in (unsorted) first-order logic can require infinite models. Even in cases where propositionalizations are finite, they can be very large and make efficient inference difficult to achieve. In this paper, we propose an approach to inference with relational theories over known and unknown objects. Like other approaches (P. Domingos, Kok, Poon, Richardson, & Singla, 2006), we exploit the fact that probabilistic inference can be framed as a weighted satisfiability problem. However, our approach does not convert relational theories into *propositional* satisfiability problems for the reasons mentioned. Instead, we define a language for expressing generative relational probabilistic (GenProb) theories over unknown objects and provide a translation from these theories to first-order weighted generative satisfiability (GenSAT) (Anonymous, under review) theories. By associating a GenProb theory with a potentially infinite probabilistic graphical model, we define a probability distribution over models of GenProb theories. The cost of the best model of a GenSAT translation corresponds to the cost of the most likely model of the GenProb problem. This equivalence permits finite maximum a posteriori inference in many cases where infinite models of a GenProb theory exist.

3.1 Limitations with Existing Systems that Unify Probabilistic Graphical Models and First Order Logic

Many of the existing approaches to integrating these two fields of probabilistic graphical models and first order logic (Domingos and Richardson 2004) require domain closure, i.e., requiring the declaring all the possible objects that can take the place of variables in the first order logic representation beforehand. For example, let us consider the first-order rule that requires every person to have a mother $\forall x$ [Person(x) $\rightarrow$ $\exists$y [Mother (y, x) $\wedge$ Person (y) $\wedge$ Female (y) $\wedge$ (x $\neq$ y)]]. Say that the

possible set of people in the domain is {Jamie, Melissa}. Then the only possible interpretation that holds is that both Jamie and Melissa are each other's mothers.

The domain closure assumption does not allow the introduction of initially unanticipated objects within the constraint system. This limitation of having to know all the objects or possible groundings of variables beforehand is problematic in several domains. Machine vision systems, for example, have to deal with occlusions and the possibility of novel objects appearing in the visual field. In the case of language, probabilistic context-free grammars, for example, permit infinite derivations of novel phrases. In general, systems that are not placed in a well-studied closed environment have the potential of offering new groundings to the existing variables and hence are beyond the scope of these approaches.

3.2 Problems due to Novel Groundings

The option of allowing novel groundings of variables in a framework that combines probabilistic and first-order representation poses two serious problems. First, since new objects or novel groundings can be introduced, the theory cannot be already propositionalized before the inference time, posing challenges for approaches based on propositionalization. Secondly, licensing the existence of new objects often leads to infinite models. For example, progressively inferring mothers will lead to an infinite model. Even in cases where propositionalizations are finite, the models that allow novel groundings can be very large, making efficient reasoning difficult.

3.3 Our Approach

Our approach to reasoning over probabilistic graphical model and relational theories is similar to existing approaches (Sang, Beame et al. 2005) in that it also exploits the fact that probabilistic inference can be framed as a weighted satisfiability problem. However current approaches translate probabilistic models to propositionalized weighted SAT constraints and perform inferences over these constraints. As described in the earlier section, the propositionalization approach fails to deal with unknown objects. Hence our approach to the problem is to convert the theory of

probabilistic and relational representation, termed GenProb, into generated weighted SAT problems, GenSAT, instead of propositionalized weighted SAT.

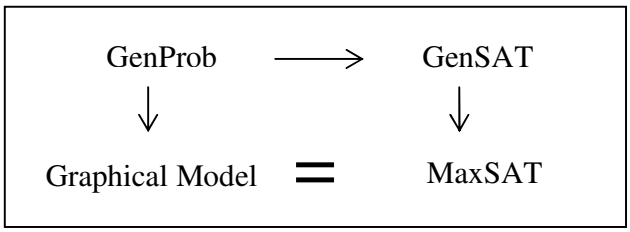

Figure 3:1 Relation between GenProb theories, GenSAT translations and models

GenSAT, as described in the previous chapter, is a weighted SAT theory that captures both relational and grounded constraints. Since the GenDPLL algorithm, which performs inferences over GenSAT, can ground constraints during inference, it is able to accommodate the licensing of new objects and also handle potentially infinite models. GenSAT and GenDPLL have been discussed in detail in the last chapter. In this chapter we focus on describing GenProb, the language that captures the combination of probabilistic and relational constraints and also licenses the creation of new groundings or objects. The thesis also proposes to define the semantics of GenProb as well as to provide translations of GenProb models to their corresponding GenSAT models.

3.4 GenProb

GenProb is a language for expressing probabilistic relational theories. The following highly simplified example theory of an airplane radar detector illustrates GenProb. "Any particular plane has a 1% chance of being within range of a radar station. The radar display generates blips that indicate a strong possibility of a plane being detected and blips that indicate a weak possibility. Strong blips are only caused by planes, whereas weak blips can be caused by noise 0.01% of the time. Planes being tracked are fitted with collision warning systems that, in the presence of other planes in range, have 90% chance of sounding an alarm that is transmitted to the radar station." The following formulae indicate the priors on a particular plane being in range and on noise:

$$True() \rightarrow (.01)\ InRange(?p)\ \wedge\ Plane(?p)$$
$$True() \rightarrow (.001)\ WeakBlip(?b)$$

The causal aspects of the theory are indicated with clauses:

$$Plane(?p) \wedge InRange(?p)$$
$$\rightarrow (.3)StrongBlip(?b)\ ,\ \ (.5)WeakBlip(?b), (.2)\ \ NoBlip(?b), ?p$$

"Of the cases where a plane is range, 30% of them will generate a strong blip, 50% of them will generate a weak blip and in 20% no blip will be generated."

$$Detect(?p1, ?p2) \wedge Plane(?p1) \wedge Plane(?p2)$$
$$\rightarrow (.9)\ TransmitAlarm(?p1), ?p1, ?p2$$

"In 90% of the cases where one plane detects another, an alarm will be transmitted." The occurrence of $?p1$ and $?p2$ after the final comma indicates that a blip licenses the existence of a plane to be inferred. The other variables are implicitly universally quantified. This will be made more precise below. The alarm detection system can be indicated thus:

$$TransmitAlarm(?p) \rightarrow AlarmSound()$$

Clauses with numbers are called causal clauses and those without number are called logical clauses. Logical clauses are hard constraints. Since blips occur in the consequent of causal clauses, they must be the effect of one of these clauses. In this case, strong blips can only be caused by planes, while weak blips can be caused by planes and by noise. Such mandatory causation is not implied by logical clauses. Mandatory causation for literals can be neutralized with a causal clause whose antecedent is True(), which (see below) is always true.

More formally, a GenProb theory is a set of causal and logical clauses. Causal clauses are of the form $C_1 \wedge ... \wedge C_n \rightarrow (p_1)E_1\ ...(p_m)E_m, ...?v_i, ...$, where $0 \leq\ p_i \leq\ 1$ and where the p_i sum to 1, each of the C_i are either literals or negations thereof, and the E_i conjunctions of literals. Each E_i conjunction is called an *effect* of the clause and each v_i is called a *posited variable*. Non-posited variables are

implicitly universally quantified. Literals are predicates with arguments that are terms. Terms that are not variables are called "objects". Logical clauses are of the form $A1 \wedge \ldots \wedge A_m \rightarrow B1 \wedge \ldots \wedge B_n$, where each conjunct is either a literal or a negation thereof. Literal *a* is a grounding of literal *b* if it is a ground literal and equivalent under an assignment of variables in *b* to objects.

3.5 Semantics of GenProb theories

In order to define semantics for a GenSAT theory, we associate it with a probabilistic directed graphical model, called its *dual graph* (DG). Each node in one of these models is a conditional probability table (CPT) as in a Bayes Network. Each node in the network can potentially have an infinite number edges directed into and out of it. We first provide a definition for the DG of a GenSAT problem and then discuss the conditions under which a probability distribution over these models is well-defined. To define the DG of a problem *P*, we first define the set of *relevant* literals for a theory. A literal is relevant for a theory if it occurs in a clause in the grounding of a theory. A clause is in the grounding of a theory if it can be derived by the repeated grounding of objects in the theory and of objects generated in previous grounding steps.

The formal definition of the grounding of a GenProb theory depends on a skolem function *s* that uniquely maps objects (that are bound to variables) to objects generated by this binding. In the example, from the last section, blip *b*, will bind to *?b* and license the inference of the existence of a plane corresponding to *b*. Technically, a skolem function maps a (potentially partial) assignment of objects to variables in a GenProb theory, to an object. In our example, for any assignment *a*, where *?b* is assigned to b, *s(a,?p) = p*. It will be convenient to abbreviate this $s_{(?b,b)}(?p) = p$. This can be read, "when *b* is assigned to *?b*, *p* is assigned to ?*p*". Even more colloquially and specifically to this example, we can say that, "*p* is the plane posited to explain *b*". As this example illustrates, the purpose of this function is to produce objects that are inferred during inference and that thus do not occur in P.

We illustrate translation with the GenSAT theory with the following four clauses.

$$Hammer(?\,h)\ \wedge\ Bell(?\,b)\ \wedge\ Hit(?\,h,?\,b)\ \rightarrow\ (.99)Ring(?\,b)$$

$$Hammer(h)$$

$$Bell(b)$$

$$Hit(h,b)$$

The translation of this theory grounds variables with constants occurring as arguments of literals in the theory.

$$Hammer(h)\ \wedge\ Bell(b)\ \wedge\ Hit(h,b)\ \rightarrow (.99)\ Ring(b)$$

$$Hammer(b)\ \wedge\ Bell(h)\ \wedge\ Hit(b,h)\ \rightarrow\ (.99)\ Ring(h)$$

$$Hit(h,b)$$

$$Hammer(h)$$

$$Bell(b)$$

Note that the second pair of clauses above are in some sense spurious since *Hammer(b)* should intuitively be false given that *Bell(b)* and that bells are not hammers. A clause indicating that hammers are not bells could prevent models with *Hammer(b)* from being true. It is also possible, though beyond the scope of this article, to eliminate such spurious clauses by extending GenProb to be a sorted logic.

To illustrate a theory over unknown objects, consider adding a clause to the effect that ringing bells generates sounds:

$$Ring(?\,b)\ \rightarrow\ GenerateSound(?\,b,?\,s),?\,s$$

Adding this clause to the GenSAT theory above would add the following two clauses to its translation:

$$Ring(b)\ \rightarrow GenerateSound(b,bSound)$$

$$Ring(h)\ \rightarrow GenerateSound(h,hSound)$$

These clauses include terms *bSound* and *hSound* which, informally, are the sounds generated by the ringing of b and of h: $s_{(?b,b)}(?s) = bSound$, $s_{(?h,h)}(?s) = hSound$. The following theory illustrates the possibility of infinite translations.

$$Mammal(a)$$

$$Mammal(?m) \wedge GiveBirth(?m, ?x) \rightarrow Mammal(?x), ?m$$

Its translation includes infinitely many clauses:

$$Mammal(a)$$

$$Mammal(aMother) \wedge GiveBirth(aMother, a) \rightarrow Mammal(a)$$

$$Mammal(aMotherMother) \wedge GiveBirth(aMotherMother, aMother) \rightarrow Mammal(aMother)$$

...

We now describe the translation more precisely in terms of a function that translates a GenProb theory onto a set of grounded propositional clauses. It is quite similar to the propositional translation (Cassimatis et al., 2009) of GenSAT, a language for expressing relational constraints over unknown objects. The translation depends on an appropriate skolem function, *s*, as described above.

A clause is in *Translation(P, s)* if it can be generated by successive steps of grounding:

$$Translation(P, s) = OneStepGrounding(P, Objects(P), s) \cup OneStepGrounding\big(P, Objects\big(Translation(P, s)\big), s\big)$$

A clause is produced in a step of grounding if it can be produced by assigning to variables objects that are either generated by a skolem function or are already in the translation.

$$OneStepGrounding(P, objects, s) = \cup_c \, ClauseGrounding(c, objects, s).$$

The grounding of a clause with respect to a set of objects is the set of clauses that can be formed by all the possible assignments of those objects to variables in the clause. For example, the grounding of $A(?x) \rightarrow B(?y), ?x$, with respect to *s* and objects *{a,b}* includes (only) the following two clauses:

$$A\left(s_{(?y,a)}(?x)\right) \rightarrow B(a)$$

$$A\left(s_{(?y,b)}(b?x)\right) \rightarrow B(b)$$

Having defined the translation of a theory, we can use the propositional literals occurring in it to define a model for that theory. Specifically, a **model** of a GenProb theory, *T*, is an assignment of propositional literals in *Translation(T)*) to truth values that is consistent with the logical constraints in that translation. A model is consistent with the logical constraints in a translation if none of the logical clauses in the translation are violated by the standard interpretation of ordinary propositional material implication.

3.6 Dual Graphs

In order to define a probability distribution over models of a GenProb theory, we introduce the notion of a dual graph (DG). Dual graphs are directed graphical models whose nodes include the propositional literals in the translation of a theory and additional literals that will be described below. The edges are related (as will be described in detail below) to clauses in the translation. Each node has a standard conditional probability distribution representing the probability of the literal being true given its parents.

The graph encodes probabilistic independence in the standard manner. Since a translation can have an infinite number of literals and since each literal can potentially participate in an infinite number of clauses, the graph can have infinite numbers of nodes, with nodes potentially participating in an infinite number of edges. DGs are defined by the merger of several sub-networks associated with the literals in the translation of a theory. Some definitions will help characterize the DG.

Caused literals are groundings of literals in the consequent of a ground causal clause. For literals $P_1(a_1, \dots), P_2(a_m, \dots.), \dots$, the literal $And(P_1, a_1, \dots, P_2, a_m, \dots, \dots)$ is their conjunction literal. An **antecedent conjunction** literal is the conjunction literal formed from the antecedents of a clause. The **consequent conjunction** literals are those conjunction literals formed from each

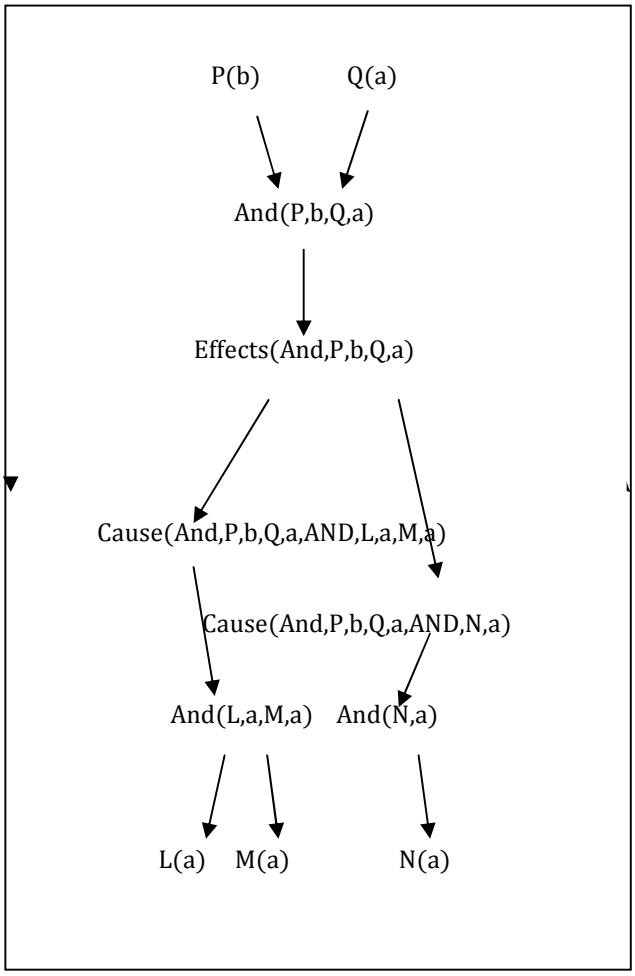

Figure 3:2 Dual graph of $P(b)$ ^ $Q(a)$ → $(.7)$ $L(a)$ ^ $M(a)$, $(.3)$ $N(a)$

effect in the literal. For every possible effect in a ground clause, we create a literal that represents the conjunction of the antecedent. An **effect literal** has predicate *Cause* and arguments: *AND* followed by the arguments of the conjunction literal,

followed by the predicate, followed by *AND*, followed by the arguments in the consequent conjunction literal. For example, the effect literal for *P(x)* → *.6 Q(x)* is *Cause(AND,P,x,AND,Q,x)*.

Several kinds of subnetworks comprise a DG. For every clause with probabilities for effect p_i, there is a **ramification network**. This network includes an edge from the conjunction literal of the clause to a multi-valued *effects literal* node – this is the only kind of non-binary node in a DG – that indicates which effect results when the left-hand-side is true. The conditional probability table is *CPT(1)* = p_i and *CPT(0) = 0*. An edge from this node is directed outward towards each effect literal for the clause. The CPT is 1 when the value corresponding to an effect is true and 0 otherwise. Figure 3.2 illustrates the ramification network and conjunction literals for $P(b) \wedge Q(a) \rightarrow (.7)\ L(a) \wedge M(a), (.3)\ N(a)$

Antecedent conjunction literals receive inputs from the individual literals in the conjunction. The CPT is 1 when all the literals are true and 0 otherwise. The consequent conjunction literals have edges directed outwards to the literals in the conjunction. The CPT for each of these is 1 when the consequent conjunction is true and 0 otherwise. In the example, the CPT for *Effects(And,P,b,Q,a)* is shown in table 3.1.

Table 3.1 Conditional Probability Table for *Effects(And,P,b,Q,a)*

And(P,b,Q,q)	Effect 1	Effect 2
0	0	0
1	.7	.3

The CPT for *Cause(And,P,b,Q,a,N,a)* is:

Table 3.2 Conditional Probability Table for *Cause(And,P,b,Q,a, N, a)*

Effects(And,P,b,Q,a)	0	1
Effect 1	1	0
Effect 2	0	1

The DG of a theory *P* is the merger of the all the ramification networks for each literal relevant to *P*. The only shared literals among these graphs are the conjunction literals and the relevant literals to *P*. The antecedent conjunction literal CPTs are identical in each ramification network and in the merged graph. CPTs for the consequent ramification literals in the merged graph have entries of 1 when at least one of the effect literal inputs is 1, and 0 otherwise. CPTs for literals only have edges entering from conjunction literals and are 1 when one of these conjunction literals is true and 0 otherwise.

A model for a GenProb problem *P* is an assignment of truth values to the values of the literals in the DG of *P* that is consistent with the logical constraints in *P*. We define a probability distribution over the models of *P* by computing a sum over the probabilities of the individual models of *P*. The probability of a model of a GenProb problem involves the product of the probabilities of the nodes in *P* given their parents. As in Bayes Networks and as will be discussed below, this distribution requires that the DG of *P* contains no cycles for that probability to be well-defined. Since the language has no explicit provision for priors, a further requirement for being well-defined is that parentless edges have only one possible value.

A probability distribution over GenProb problems with models *M* that is consistent with logical constraints in *M* can be defined thus:

$$P(m) = \frac{f(m)}{\sum_{m_i \in M} f(m_i)}$$

where for literals *L* in *m* and where m_l is a model *m* without an assignment for l:

$$f(m) = \begin{cases} \prod_{l_i \in L} P(l_i | m_l), when\ finitely\ many\ literals\ are\ true\ in\ L \\ 0,\ otherwise \end{cases}$$

For many infinite models, the product in *f* converges on zero in the limit. One class of cases where this is guaranteed to happen involves *decreasing probability theories*. In a decreasing probability theory, each object that is the argument of a literal of the DG is an argument of a literal whose probability given the rest of the DG is less than 1. Although some problems are not naturally cast as decreasing probability theories, e.g., arithmetic, there are many cases where each potential new

object does lead to increased uncertainty. For example, in radar interpretation, each blip can be caused by a plane or by noise; in most gesture recognition situations, a person is never certain to produce a specific gesture; any probabilistic context-free grammar whose phrases can all be generated using more than a production rule is a decreasing probability theory.

In decreasing probability theories, therefore, it is natural to define $f(model) = 0$ when there are infinite numbers of literals assigned to true. $f(model)$ has finite nonzero addends because every CPT in the model is zero when all the parent nodes are zero. Since only finite numbers of nodes can have nonzero parents, there are only finite nonzero addends.

The resulting probability distribution is well defined in many, but not all, cases. As in Bayesian Networks, in order for f to be well-defined, the DG cannot contain directed cycles. When involved in such a cycle, the probability of a particular node value would be defined in terms of itself. DGs of GenProb theories contain no directed cycles in (but not only in) the following circumstance: no predicate appears in its own antecedent predicate closure. The antecedent closure for a predicate, A, contains the union of all predicates in the antecedent of clauses that have A in the consequent together with the predicate closure of those predicates. In other words, the antecedent closure for A contains all the predicates that can be involved in causal chains supporting A. If a literal is involved in a cycle, then its predicate will clearly be in its own antecedent predicate closure. Thus, when a predicate does not appear in its antecedent closure, there can be no cycles in the DG.

As mentioned earlier, the parentless nodes in the DG must have only one possible value. One way to accomplish this that proves useful in practice is to include a causal clause whose antecedent is *True()* for each type of literal that is not in an effect of a causal clause.

Proposition 1. GenProb theories have a well-defined probability distribution if no predicate appears in its own antecedent closure and when for each type of literal l not in the effect of another clause there is a clause, $True() \rightarrow (p)\ l$, where $p > 0$.

Although we would like to find stronger conditions for a well-defined probability distribution – the conditions in Proposition 1 exclude some of our examples, for instance – some important classes of theories do meet Proposition 1's conditions, however. In that absence of such conditions, however, we must rely on the fact that theories without directed cycles in their DG will admit well-defined probability distributions.

Proposition 2. GenProb theories have well-defined probability distributions if their DGs do not contain directed cycles and if for each type of literal *l*, not in the effect of another clause, there is a clause, *True()* → *(p) l, where p > 0*.

This proposition will be useful in some specific, but important cases. Although we do not prove it here, for example, the GenProb translation (Murugesan & Cassimatis, 2009) of a probabilistic context-free grammar intuitively will have a DG without cycles because no specific phrase can generate itself. (Of course, categories of phrases can generate phrases with the same category as themselves, e.g., NP → NP + PP).

Note that the above conditions on well-definedness pertain only to causal clauses. Logical clauses only serve to rule out models. For this reason and because problems with cyclicity generally do not obtain when all probabilities are zero or one, there is no need to impose constraints on the cyclicity of logical constraints in a GenProb theory in order for there to be a well-defined probability distribution over its models.

Approaches that involve probabilistic inference over infinite numbers of objects (Kersting & De Raedt, 2001; Milch, Marthi, Sontag, Russell & Ong, 2005; Brian Milch et al., 2005; Muggleton, 1996; Richardson & Domingos, 2007), typically require that a node be directly connected with only finitely many other nodes in a graph. Infinite parents in GenProb DGs only occur for conjunction and literal nodes since all other nodes are defined to have only a fixed number of parents. These nodes have CPTs that are nonzero only when one of the parents is true and are thus

well-defined even in the infinite case. In models where there are infinite parents that are true for a node, the probability of the model, by the definition of *f*, is zero.

The following is simple example of a theory where infinite parents for a node arise. "A parent with a particular gene, X, is likely to pass it on to its offspring and that some, but many fewer, instances of a gene occur by chance. Genes generate a symptom Y, which can also occur through contagion from others with that symptom. Finally, individual *a*, has the Y phenotype."

The following clauses formalize this theory in GenProb:

$$Parent(?p, ?c) \wedge GeneX(?p) \rightarrow (.2)\ GeneX(?c), ?p$$

It is read as "if $?p$ is the parent of $?c$ and $?p$ has Gene X, then there is a 20% chance $?p$ will have passed X to $?c$."

$$True() \rightarrow (.000001)\ GenX(?c)$$

"In one out of a million cases, a person can have Gene X without any other cause, i.e., through mutation."

$$Y(?a) \rightarrow (.1)\ SpreadYTo(?a, ?p) \wedge Y(?p)$$

"Someone with Y has a 10% chance of spreading it to someone else."

$$GeneX(?p) \rightarrow Y(?p)$$

"Gene X always generates Y"

$$Y(a)$$

"a has Y"

The DG for this theory is infinite because one can acquire Y through one's parent, who can acquire it from his parent and so on. However, the probability of each inheritance is less than one and thus the probability of a model with an infinite number of ancestors with the gene is zero. Models with nonzero probability are

possible. For example, the model where *Y(a)* and *GeneX(a)* are true, and all other literals are false, has finite probability.

3.7 Inference

As in the case of some other probabilistic frameworks (P. Domingos et al., 2006), inference in GenProb theories can be formulated as a weighted satisfiability problem. However, since GenProb theories can have infinite models, they cannot be straightforwardly translated to ordinary MaxSAT problems. Instead, we must translate them to a weighted satisfiability framework that can deal with unknown objects. Specifically, in this section we show that a GenProb theory can be translated into a generative weighted satisfiability (GenSAT) problem such that cost of a model, *m*, for the GenSAT problem corresponds to *f(m)*.

GenSAT theories are relational weighted satisfiability problems. They are expressed in a language for encoding weighted and logical constraints that license the positing of new objects. Their syntax has many similarities with that of GenProb theories. The following simple example illustrates the main aspects of the language. It involves constraints that state that both a telephone (whose ringer is on) receiving a call and the striking of a bell lead to a ringing sound and that a phone ringer is only on when the phone's battery is not empty.

$$BellStrike(?x, ?t) \rightarrow (10)\ RingSound(?t), ?x$$

$$PhoneGetsCall(?p, ?t) \wedge RingerOn(?p, ?t) \rightarrow (7)\ RingSound(?t), ?p$$

$$RingerOn(?p, ?t) \rightarrow EmptyBattery(?p, ?t)$$

The numbers "10" and "7" are the weights on the constraints represented by the clauses. Terms beginning with '?' are variables. Variables interpreted after the comma are called *posited* variables and indicate the potential of an object to be posited. For example, ?x in the second constraint indicates that if a ring sound occurs, then a bell (as of yet unknown) being struck may have caused it. Non-posited variables are implicitly universally quantified. Clauses with weights are causal clauses and those without are hard constraints.

Each GenSAT theory can be translated into a potentially infinite set of weighted MaxSAT constraints. The atoms over which these constraints hold also include the set of relevant literals for the GenProb. A GenSAT model is an assignment of truth values to these literals. Each model has a cost, which is the sum of the cost of all the ground constraints that are broken by it.

The MaxSAT translation of GenSAT models imposes a "mandatory causation" constraint on literals. Literals that occur in the consequent of a causal clause must be "caused" by some causal clause constrained to be true. Thus, if *P(a)* → *10 R(a)* and *Q(a)* → *15 R(a)* are the only two causal clauses with *R(a)* in the antecedent, any model where *R(a)* is true must have one of *P(a)* or *Q(a)* being true.

Our approach is to translate a GenProb problem into a GenSAT problem such that probability of a GenProb model can be discerned from the cost of the corresponding GenSAT model. The minimal-cost GenSAT model corresponds to the most likely model of the GenProb problem. Since at least one complete algorithm (Cassimatis et al., 2009) exists for GenSAT problems, this mapping therefore provides a method of inference for GenProb theories. We associate a GenProb theory *P* with a GenSAT theory *S* so that the total cost of the constraints broken in the best model of *S* is equivalent to the negative log of the probability of the best model of *P*. Since the equation for the probability of the best model of *P* is a product, we use its log so that the resulting sums can be more easily associated the addends making up the sum of the broken constraints in models of *S*. Specifically:

$$\ln f(m) = \sum_{l \in L} \ln P(l|m_l)$$

Maximizing this quantity is equivalent to minimizing the following quantity:

$$-\ln f(m) = -\sum_{l \in L} \ln(P(l|m_l)$$

Note that each addend corresponds to a literal in the DG. The translation creates GenSAT constraints whose cost equals each addend. For example, if $P(l|m_l) = .75$

in a DG, then a constraint is broken in the corresponding model for the GenSAT translation whose cost is *-ln .75*.

Let *C(i,o)* be the cost of the constraints broken in a model given the values of *i* and *o*, where there is only one edge leading into *o,* that edge is *i* and whose CPT is p_v for possible value *v* if *i* is true and 0 otherwise. This is the kind of CPT for the effect literal nodes in a ramification network. *C* must generate the following costs:

$$C(1,1) = -\ln P(i = 1|o = 1) = -\ln p_v$$
$$C(1,0) = -\ln P(i = 0|o = 1) = -\ln(1 - p_v)$$
$$C(0,1) = -\ln P(i = 1|o = 0) = -\ln 0 = \infty$$
$$C(0,0) = -\ln P(i = 0|o = 0) = -\ln 1 = 0$$

For clauses with antecedent *A*, consequent effects, C_i and a list of posited variables *V*, the following GenSAT constraints, where *conj(X)* is the conjunction literal for literals *X* and the *effect(A,*c_i*)* are the effect literals for the clause, will lead to these costs:

$$conj(A) \rightarrow (ln\ p)\ \ \neg effect(A, C_i), V$$
$$conj(A) \rightarrow (ln\ p)\ \ \neg effect(A, C_i), V$$
$$\neg conj(A) \rightarrow \neg effect(A, C_i)$$
$$\neg effect(C_1) \wedge \neg effect(C_n) \rightarrow \neg conj(A)$$

Since the entries in CPTs for all other nodes are 1 or 0, they can be captured with straightforward logical constraints.

The translation of a GenProb theory into a GenSAT theory is the union of all the constraints capturing the conditional probability table of the nodes in the DG together with the constraints defining the conjunction and effect literals.

As in Bayes Networks, implicit in DGs is that an inner node's value only occurs under or is "caused by" certain configurations of the values of the nodes directed towards it. This is automatically captured by GenSAT "mandatory causation". For example, if *A* and *B* are the only two inputs to node *C*, there will be two causal clauses in the translation with *C* in the consequent. As in the DG, if *A* and *B* are false in a model of the GenSAT translation of a GenProb theory, then so must be *C*.

We have shown that for each literal l in the joint probability equation f, there will be a cost broken in the GenSAT translation of it that corresponds to the probability of l given its parents. Since these are the only constraints broken, the cost of a model, m_S, for a GenSAT translation of a GenSAT problem P, corresponds to $f(m)$, where m_P is a model of P.

Proposition 3. For model m_p of a GenSAT theory P and model m_S of its GenSAT translation S:

$$-\ln f(m_P) = cost(m_S).$$

This fact enables algorithms for finding GenSAT models to make inferences about theories expressed in GenProb. The GenDPLL algorithm (Cassimatis et al., 2009) finds minimal cost models for "increasing cost theories". Increasing cost theories have models whose posited objects are guaranteed to participate in broken constraints that have net positive costs. Since each causal clause in a decreasing probability GenProb theory is translated into GenSAT constraints that are guaranteed to generate positive costs, GenDPLL will therefore identify solutions to translations of decreasing probability GenSAT problems.

3.8 Implementation

This work has been motivated by the Cognitive Substrate Theory (Cassimatis 2006) and implemented in Polyscheme cognitive architecture (Cassimatis 2005).

3.8.1 Motivation

The cognitive substrate theory proposes that a few basic conceptual elements underlie all of cognition and that more complex cognitive tasks can be expressed in terms of these basic cognitive functionalities. Examples of how different conceptual fields can be reduced to the same basic cognitive substrate have been shown (Cassimatis 2004, Bugajska et al., 2006, Bello et al., 2006). This substrate includes reasoning and representation abilities for time, space, causality, identity, part-hood and desire.

However the substrate theory only outlines the behavior required (i.e.) stays at the Marr's computational level of abstraction. Polyscheme cognitive architecture on the other hand is an algorithmic implementation of the substrate theory, for there may be several ways of implementing the substrate theory of which Polyscheme is one example. Hence, Polyscheme belongs to Marr's algorithmic level of abstraction.

Research in substrate theory has already shown that language can be characterized in terms of the basic substrate elements. Furthermore, research (Cassimatis 2004, Murugesan and Cassimatis 2006) has shown how linguistic grammars can be implemented in Polyscheme. One of the most challenging aspects of language is uncertainty and Polyscheme was unable to handle reasoning with uncertainty before. Hence, implementing reasoning with uncertainty in a framework like Polyscheme has became a motivating factor in achieving NLP. Also, Polyscheme and the substrate support the functionality of inferring unknown objects. This combination of supporting the inference of unknown objects while reasoning with uncertainty is fairly new to the field and has required novel research.

3.8.2 DPLL Specialist

GenProb, GenSAT and GenDPLL have been implemented within the Polyscheme cognitive architecture. Polyscheme architecture supports a number of independent specialists which share their opinion on a piece on knowledge (a proposition) using the focus of attention mechanism. We have implemented a DPLL specialist which takes a number of GenProb, GenSAT, rules and logical rules during initialization.

```
<?xml version="1.0" encoding="UTF-8"?>
<model name="DPLL Model" outputfile="DPLLOutput.txt"
print="true">
<specialist
name="perception.TemporalPerceptionSpecialist">
<step time="1">
FoundSentence(true, t1, t3, E, R)
NP(NP-Phrase1, t1, t2, E, R)
```

```
Exists(NP-Phrase1, E, R)
VP(VP-Phrase2, t2, t3, E, R)
Exists(VP-Phrase2, E, R)
</step>
</specialist>

<specialist name="rules4.DpllSpecialist">
<rule flags="[false,false,true][true,true,false]">
FoundSentence(true, t1, t3, E, ?w)
0.98> :
S(?phrase, t1, t3, E, ?w)
</rule>

<rule
flags="[true,true,false][true,true,false][true,true,fals
e]" isMutualEx="true">
PartOf(?part, ?obj1, E, ?w ) + PartOf(?part, ?obj2, E,
?w )
 ==> :
Same(?obj1, ?obj2, E, ?w)
</rule>
</specialist>
</model>
```

A Polyscheme model has input defined for several specialists. In this example, temporal perception specialist and dpll specialist both have input. The format of input to the dpll specialist is a number of embedded rules with header flags and body consisting of the constraint itself. The behavior of the rules is described in the next section.

3.8.3 Behavior of DPLL Specialist

During initialization, the GenDPLL specialist converts GenProb rules to corresponding GenSAT rules. A GenProb rule is converted to a GenDPLL rule by replacing the probability with costs and adding an intermediate layer of Cause propositions. An example of converting a GenProb rule to its corresponding GenSAT rules can be shown.

GenProb rule

```
<rule flags="[true,true,false] [false,false,false]">
A(?y)
0.x> :
B(?y)
</rule>
```

Corresponding GenSAT rules

```
<rule flags="[true,true,false] [false,false,false]">
A(?y)
-log(0.x)> :
Cause(A, B, ?y)
</rule>

<rule flags="[true,true,false] [false,false,false]">
A(?y)
-log(1-0.x)> :
NOT Cause(A, B, ?y)
</rule>

<rule flags="[true,true,false] [false,false,false]">
Cause(A, B, ?y)
==> :
```

```
B(?y)
</rule>
```

3.8.4 One-Step Grounding

One-step grounding is a process unique to GenSAT. In Polyscheme, one-step grounding is implemented by the leveraging the focus of attention mechanism. At every Polyscheme cycle, a proposition is chosen for focus. GenSAT treats this proposition as the new knowledge introduced to the system, and tries to ground constraints using this focal proposition. Furthermore, Polyscheme offers increased flexibility in controlling the one-step grounding process by introducing flags for GenSAT constraints.

Each rule has an optional flag whose number of entries should match the number of terms in the rule, where each entry is a Boolean triplet. Corresponding to each propositional term is the Boolean triplet indicating "should match, should posit and should retrieve" in that order. The focal proposition is allowed to ground a proposition only if its corresponding should match Boolean value is true. If should posit Boolean value is set to true, then that proposition is grounded by positing new objects for the unbound variables of that proposition. Similarly, should retrieve Boolean indicates if the proposition is allowed to be grounded by propositions that have been focused on in the past.

Once a constraint can be completely grounded by using the focal proposition and applying the appropriate flags, then this grounded constraint is now added to network of constraints. While adding the new grounded constraint, allCausesKnown propositions that enforce mandatory cause constraint have to be updated.

For example if the completely grounded constraint to be added is

```
<rule flags="[true,true,false] [false,false,false]">
Cause(A, B, blast1)
==> :
B(blast1)
</rule>
```

And the existing mandatory causation rule for B(blast1) is

```
<rule>
NOT Cause(C, B, blast1) + AllCausesKnown(B, blast1, 1)
==> :
NOT B(blast1)
</rule>
```

The new mandatory causation rule for B(blast1) would now be

```
<rule>
NOT Cause(C, B, blast1) + NOT Cause(A, B, blast1)
 + AllCausesKnown(B, blast1, 2)
==> :
NOT B(blast1)
</rule>
```

And the truth value of `AllCausesKnown(B, blast1, 1)` is set to false and the new all causes know prop `AllCausesKnown(B, blast1, 2)` is set to likely.

3.8.5 Polyscheme Worlds

Polyscheme has built in support for alternate worlds which is a concept very similar to Talmy's mental spaces (Talmy 1988). Every piece of knowledge, proposition, in Polyscheme is tagged with the world in which it exists. Also inheritance between worlds is handled in Polyscheme.

Polyscheme has a special world designated with the name R which stands for real world. R world is unique in that it theoretically reflects the real world and there are no other assumptions. All other Polyscheme worlds inherit knowledge from R and have a list of propositions with truth values that override the values in R, called the world's basis. Polyscheme is thus able to represent hypothetical and even counterfactual worlds.

3.8.6 GenDPLL Implementation

GenDPLL is implemented using Polyscheme's worlds. GenDPLL searches by branching on each uncertain literal by assigning it true in one search space vs. assigning it false in the other. Similarly two Polyscheme worlds are created at every branch point in GenDPLL. For examples if W is the existing Polyscheme world and A is now the new uncertain literal to branch upon then the following two Polyscheme worlds are created.

```
WA with basis of W + A being true
W not A with basis of W + A being false.
```

Polyscheme also associates each world with a cost. Thus a Polyscheme world inheriting from another, as in the case of WA inheriting from W, inherits the cost of the parent world. Hence, by implementing DPLL search spaces as Polyscheme worlds, each model or solution automatically has a cost associated with it.

3.9 Summary of GenProb

Relational probabilistic theories are an effective means of compactly representing and efficiently making inferences in many situations. Unknown objects and infinite domains can lead to several formal and computational complications using an approach that propositionalizes a relational theory before inference. By expressing theories when using the GenProb language and translating them into graphical models, a probability distribution over models of these theories can be well-defined. Translations of GenProb theories to GenSAT models enable correct inference, even in many cases that involve models of infinite size. The final step in our goal, which is to reason over both syntax and semantics represented in the GenProb formalism, is to encode the implicit constraints of Probabilistic Context Free Grammar (PCFG) in the GenProb language. The next chapter identifies the implicit constraints of PCFG and proposes to encode these constraints in the GenProb formalism.

4. PCFG

The languages GenProb and GenSAT have been defined and the reduction of GenProb theory to GenSAT constraints has been proposed in the previous chapters. The final step of integrating syntax and semantics is to show that PCFG can be encoded in the proposed GenProb formalism. In this chapter, we identify the implicit constraints of PCFG and propose the encoding of these constraints in the GenProb language.

4.1 Reasons for Choosing PCFG

An important aspect of general intelligence is that the same method can be applied to various problems spanning different domains. It is believed that several commonalities underlie the various domains of cognition and some of them have been pointed out by the theory of the Cognitive Substrate (Cassimatis, 2006). These include temporal ordering, part hierarchies, generative processes and categories. Probabilistic Context Free Grammars (PCFG) is a formalism that has been widely used to model these phenomena in various domains like vision, RNA folding and Natural Language Processing. Hence improving the coverage of PCFG and integrating PCFGs with a general probabilistic inference framework is a step towards achieving Artificial General Intelligence (AGI).

4.2 Probabilistic Context Free Grammar

Jointly reasoning over PCFG and other constraints is enabled by expressing PCFG problems in the GenProb language. A PCFG has a set of rules, R, where each rule is of the form:

$$
\begin{aligned}
X \rightarrow \quad & (prob_1) u_{11} u_{12} \ldots u_{1m1} \\
& \mid (prob_2) u_{21} u_{22} \ldots u_{2m2} \\
& \ldots \\
& \mid (prob_n) u_{n1} u_{n2} \ldots u_{nmn}
\end{aligned}
$$

where the antecedent X on the left hand side (LHS) of the rule is called a non-terminal. The rule is called a production and is described as the non-terminal X generating the right hand side (RHS) symbols. A symbol that does not generate any further symbols i.e. never occurs on the LHS of a rule is called a terminal.

The rule also captures the probability that the non-terminal generates a particular set of symbols like $u_{11}u_{12} \ldots u_{1m1}$ or $u_{21}u_{22} \ldots u_{2m2}$ through the numbers $prob_1$ and $prob_2$ respectively. The sum of all the probabilities is 1.

$$\sum_{i=1}^{n} prob_i = 1$$

A grammar G generates a language L(G) using the PCFG rules in R. The functionality of a parser P for a given string (I) is to determine whether and how this string (I) can be generated from the grammar (G) (i.e., to determine if (I) is a member of L(G)). There are several implicit constraints in the generation of a language. Our aim is to formalize and explicitly encode these constraints in the GenProb language.

4.3 Implicit constraints of PCFG

As described earlier, the implicit constraints enforced by PCFG have to be identified. These constraints are listed in this section and they include order of rule arguments, structure of parse tree nodes and mandatory manifestation of non-terminals among others.

4.3.1 The Order Constraint

Probabilistic constraints in most language are generally order independent with regard to both the order of the input and the order of the terms in their constraints. However the language generated by G depend on several ordered components including the order of the list of terminals in the string (I) and the order of RHS components in a PCFG rule.

4.3.1.1 Ordering of Input

Let the input I, say $a_1 a_2 \ldots a_n$, be the ordered sequence of input terminals for which the parse has to be determined. The general notion of ordering of events can be broken down into two step: capturing the time of occurrence of an event (both start and end points) and establishing relations between these time of occurrences. The constraints of $I(a_1 a_2 \ldots a_n)$ is captured using the following grounded propositions.

$$Occur(a_1)$$
$$IsA(a_1, A_1)$$
$$StartTime(a_1, t_0)$$
$$EndTime(a_1, t_1)$$
$$Meets(t_1, t_1)$$

$$Occur(a_2)$$
$$IsA(a_2, A_2)$$
$$StartTime(a_2, t_1)$$
$$EndTime(a_2, t_2)$$
$$Meets(t_2, t_2)$$

...

$$Occur(a_n)$$
$$IsA(a_n, A_n)$$
$$StartTime(a_n, t_{n-1})$$
$$EndTime(a_n, t_n)$$

In these propositions $A_1 A_2 \ldots A_n$ represent the part of speech of the words $a_1 a_2 \ldots a_n$ respectively.

4.3.1.2 Ordering of Rule Arguments

A typical PCFG rule(R) is of format:

$$X \rightarrow \quad (prob_1)u_{11}u_{12} \ldots u_{1m1}$$
$$| \, (prob_2)u_{21}u_{22} \ldots u_{2m2}$$
$$\ldots$$
$$| \, (prob_n)u_{n1}u_{n2} \ldots u_{nmn}$$

The rule is dependent on the order of the u_i symbols. According to the definition of R, u_i symbols can be either terminals or non-terminals. The same ordering technique used to order the input terminals I can be used to order RHS components of R. However it is to be noted that this scheme also requires associating non-terminal symbols with the time of their occurrence. Hence the non-terminal X on the LHS is also associated with the entire time interval of all the consequents. Therefore, a typical PCFG rule can be broken down into components captured as follows.

$$Occur(?\,xobj) \wedge$$
$$IsA(?\,xobj, X) \wedge$$
$$StartTime(?\,xobj, ?\,t_0) \wedge$$
$$EndTime(?\,xobj, ?\,t_{\mathrm{n}})$$

$$\rightarrow (prob_1)$$

$$Occur(?\,u_1obj) \wedge$$
$$IsA(?\,u_1obj, u_1) \wedge$$
$$StartTime(?\,u_1obj, ?\,t_0) \wedge$$
$$EndTime(?\,u_1obj, ?\,t_1) \wedge$$
$$Meets(?\,t_{1,}, ?\,t_1) \wedge$$

$$\ldots$$
$$Occur(?\,u_{1m1}obj) \wedge$$
$$IsA(?\,u_{1m1}obj, u_{1m1}) \wedge$$

$$StartTime(?u_{1m1}obj, ?t_{n-1}) \wedge$$
$$EndTime(?u_{1m1}obj, ?t_{n})$$

4.3.2 Parse Tree Nodes

Every node in the parse tree has to be marked with the time it spans. The same terminal might occur twice in the input I, even though the nodes corresponding to the two symbols are different. For example, in the phrase "the man in the park", the word "the" (terminal) occurs twice; however, the parse tree node corresponding to the two "the" words different. Similarly every non-terminal in the parse tree has a distinct time of occurrence, even though the same non-terminal can occur several times within the parse tree.

4.3.3 Unique Dominator

Another implicit constraint of L(G) is the unique parent relationship. Every node can have only one parent creating a strict parse tree and disallowing a multi-tree. This unique parent relationship is captured in GenProb language by introducing part-hood associations. Every node belongs or is a part of its parent node, and a node cannot be a part of more than one parent.

$$PartOf(?childNode, ?parentNode_1) \wedge$$
$$PartOf(?childNode, ?parentNode_2) \wedge$$
$$NOT\ Same(?parentNode_1, ?parentNode_2)$$
$$\Rightarrow$$
$$FALSE$$

4.3.4 Creation of New Objects

It is to be noted that when the right hand side of a PCFG rule is matched, i.e. when the terminals occur next to each other in the correct order, the non-terminal or the left hand side of the PCFG rule is posited. Hence, for every rule matching a new parse tree node is created. Therefore, the GenProb language will also require the capability of creating or inferring new parse tree nodes.

4.3.5 Alternative Manifestations

In the PCFG grammar G, for every non-terminal symbol all the alternate options are listed with their respective probabilities.

$$\begin{aligned} X \rightarrow \quad & (prob_1)u_{11}u_{12} \ldots u_{1m1} \\ & \mid (prob_2)u_{21}u_{22} \ldots u_{2m2} \\ & \ldots \\ & \mid (prob_n)u_{n1}u_{n2} \ldots u_{nmn} \end{aligned}$$

A particular non-terminal node can only generate one of the options.

4.3.6 Every Node has a Parent

In PCFG since the cause of every node is the parent node generating it, the constraint that at least one parent of every node should be true should be captured in GenProb language. Hence there can be no node that is unconnected to the parse tree.

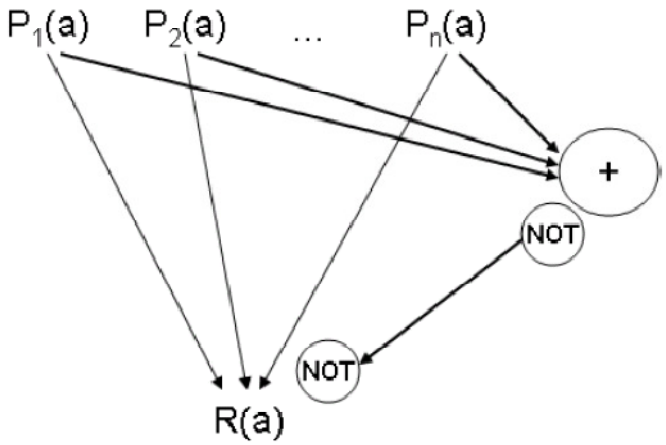

Figure 4:1 Mandatory Parent Rule

$R(a)$ cannot exist without at least one of its parents

$$NOT\ P1(a) \wedge NOT\ P2(a) \wedge \ldots NOT\ Pn(a) \wedge \Rightarrow NOT\ R(a)$$

4.3.7 Start Symbol

The one node in the parse tree that does not have a parent is the start node S, and every other node in the tree has to be connected via other parents to this start node.

$$TRUE$$
$$\Rightarrow$$
$$IsA(?obj, S) \wedge$$
$$Occur(?obj) \wedge$$
$$StartTime(?obj, Istrt) \wedge$$
$$EndTime(?obj, Iend)$$

4.3.8 Mandatory Manifestation

All the leaf nodes in a parse tree have to be terminals. This axiom ensures that every non-terminal in a parse tree should generate a string based on R, which we call the mandatory manifestation of non-terminals. A parse tree that does not satisfy the mandatory manifestation constraint is an invalid tree as shown in figure 4.1.

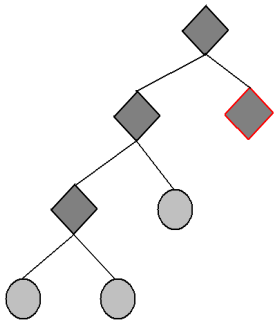

Figure 4:2 Mandatory Manifestation Rule Violations

Diamonds represent the non-terminals, one of which is the non-terminal leaf node that violates the mandatory manifestation rule

This constraint of saying that among all the possible generation of a non-terminal at least one of them should hold true is should be captured.

$$Occur(?obj) \wedge$$
$$NOT\ AtleastOneEffect(?obj)$$
$$\Rightarrow$$
$$FALSE$$

Say there are 2 productions that can be generated from a non-terminal say X.

$$X \rightarrow$$
$$(0.75)a$$
$$|\ (0.25)Y\ Z$$

The rules that posit $AtleastOneEffect$ of the non-terminal X look like:

1.

$$Occur(?aobj) \wedge$$
$$IsA(?aobj, a) \wedge$$
$$StartTime(?aobj, ?tstrt) \wedge$$
$$EndTime(?aobj, ?tend) \wedge$$

$$Occur(?xobj) \wedge$$
$$IsA(?xobj, X) \wedge$$
$$StartTime(?xobj, ?tstrt) \wedge$$
$$EndTime(?xobj, ?tend)$$
$$\Rightarrow$$
$$AtleastOneEffect(?xobj)$$

2.

$$Occur(?yobj) \wedge$$
$$IsA(?yobj, Y) \wedge$$
$$StartTime(?yobj, ?tstrt) \wedge$$
$$EndTime(?yobj, ?tmid1) \wedge$$

$$Occur(?zobj) \wedge$$
$$IsA(?zobj, Z) \wedge$$

$$StartTime(?zobj, ?tmid2) \wedge$$
$$EndTime(?zobj, ?tend) \wedge$$

$$Meets(?tmid1, ?tmid2) \wedge$$

$$Occur(?xobj) \wedge$$
$$IsA(?xobj, X) \wedge$$
$$StartTime(?xobj, ?tstrt) \wedge$$
$$EndTime(?xobj, ?tend)$$

$$\Rightarrow$$
$$AtleastOneEffect(?xobj)$$

4.4 Implementation of GenProb Rules for Representing PCFG

The process of creating GenProb rules for a PCFG rule can be automated. However, the number of GenProb rules created for a PCFG rule is not constant and depends on (1) the number of non-terminals in LHS that are expanded by PCFG rules and (2) the number of alternate options in each PCFG rule.

4.4.1 GenProb Rules Created per Grammar

To begin with, there are a few GenProb rules that are created once for the entire grammar. For example to encode that the start symbol of a PCFG is a sentence one GenProb rule is introduced. Also, 3 GenProb rules are used to represent the unique parent constraint.

```
<rule flags="[false,false,true][true,true,false]">
FoundSentence(true, t1, t8, E, ?w)
0.98> :
S(?phrase, t1, t8, E, ?w)
</rule>
```

```
<rule
flags="[true,true,false][true,true,false][true,true,fals
e]" isMutualEx="true">
PartOf(?part, ?obj1, E, ?w ) + PartOf(?part, ?obj2, E,
?w )
 ==> :
Same(?obj1, ?obj2, E, ?w)
</rule>

<rule flags="[true,true,false][true,true,false]">
NOT Exists(?obj, E, ?w)
 ==> :
NOT PartOf(?part, ?obj, E, ?w )
</rule>

<rule flags="[true,true,false][true,true,false]">
NOT Exists(?obj, E, ?w)
 ==> :
NOT PartOf(?obj, ?whole, E, ?w )
</rule>
```

4.4.2 GenProb Rules Generated per Part of Speech

There are 3 GenProb rules created per PCFG terminal being expanded. For example, the following three rules are introduced to capture the non-terminal S being expanded.

```
<rule flags= "[true,true,false][true,true,false]">
NOT AtleastOneEffect(S,?startOf1Child, ?endOf2Child, E,
?w)
 ==> :
NOT CreateSPhrase(?startOf1Child, ?endOf2Child, E, ?w)
```

```
 </rule>

<rule        flags="[true,false,false][false,true,false]"
isMutualEx="true">
NOT CreateSPhrase(?startOf1Child, ?endOf3Child, E, ?w)
 ==> :
 NOT S(?Phrase-, ?startOf1Child, ?endOf3Child, E, ?w)
 </rule>

<rule flags="[false,false,true][true,true,false]">
S(?Phrase-, ?startOf1Child, ?endOf3Child, E, ?w)
 ==> :
CreateSPhrase(?startOf1Child, ?endOf3Child, E, ?w)
 </rule>
```

4.4.3 GenProb Rules Generated per PCFG Rule Option

A PCFG rule has several options, for example the following PCFG rule has 2 options NP VP and MD NP VP.

S → (0.75) NP VP |
(0.25) MD NP VP

For each PCFG option 5 rules are generated: 3 rules that capture the option, one GenProb rule that enforces the mandatory manifestation constraint, and the last GenProb rule that posits the part of relation between the parent and child phrases which helps uphold the unique parent constraint.

```
<rule flags="[false,false,false][true,true,false]">
CreateSPhrase(?startOf1Child, ?endOf2Child, E, ?w)
0.50> :
```

```
EffectOfNPDueToRule(SAndNPVP,              ?startOf1Child,
?endOf2Child, E, ?w)
</rule>

<rule flags="[false,false,false][true,true,false]">
EffectOfNPDueToRule(SAndNPVP,              ?startOf1Child,
?endOf2Child, E, ?w)
==> :
EffectOfS(?phrase1,  ?phrase2,  SAndNPVP,  ?startOf1Child,
?endOf1Child, ?endOf2Child, E, ?w)
</rule>

<rule flags="[false,false,true][true,true,false]">
EffectOfS(?phrase1,  ?phrase2,  SAndNPVP,  ?startOf1Child,
?endOf1Child, ?endOf2Child, E, ?w)
 ==> :
AndNPVP(?phrase1,        ?phrase2,        ?startOf1Child,
?endOf1Child, ?endOf2Child, E, ?w)
</rule>
```

Mandatory manifestation implementation

```
<rule flags="[true,true,false][true,true,false]">
EffectOfS(?phrase1,  ?phrase2,  SAndNPVP,  ?startOf1Child,
?endOf1Child, ?endOf2Child, E, ?w)
 ==> :
AtleastOneEffect(S, ?startOf1Child, ?endOf2Child, E, ?w)
 </rule>
```

Unique parent implementation

```
<rule
flags="[true,true,false][true,true,false][false,false,fa
lse][false,false,false]" isMutualEx="true">
EffectOfS(?phrase1, ?phrase2, SAndNPVP, ?startOf1Child,
?endOf1Child, ?endOf2Child, E, ?w) +
S(?biggerPhrase, ?startOf1Child, ?endOf2Child, E, ?w)
 ==> :
 PartOf(?phrase1, ?biggerPhrase, E, ?w)
 + PartOf(?phrase2, ?biggerPhrase, E, ?w)
 </rule>
```

4.4.4 Alternate Options

A number of rules are introduced to enforce that only one of the alternate options holds true, i.e. if there are n options in a PCFG rule then (n-1) + (n-2) + … + 1 number of rules are introduced. In the case of

S → (0.75) NP VP |
(0.25) MD NP VP

the following GenProb rule is introduced.

```
<rule flags="[true,false,false][true,false,false]">
EffectOfSDueToRule(AndMDNPVP, ?startTime, ?endTime, E,
?w)
 ==> :
NOT EffectOfSDueToRule(AndNPVP, ?startTime, ?endTime, E,
?w)
</rule>
```

4.4.5 Number of GenProb Rules for a Sample PCFG Grammar

To summarize, the number of GenProb rules generated for the sample PCFG grammar shown here can be calculated.

S → (0.50) NP VP |
(0.35) X Y |
(0.15) MD NP VP

VBD → (0.75) X Y |
(0.25) MD VBD

NP → (0.03) POS NN |
(0.07) X Y |
(0.05) PRP NP |
(0.15) JJ NP |
(0.45) DT NP |
(0.25) NP PP

VP → (0.10) VP NP |
(0.08) MD VP |
(0.11) VBD PP |
(0.14) VBD JJ |
(0.10) VB NP |
(0.24) VP PP |
(0.07) MD VB |
(0.11) VP NP NP |
(0.05) VB NP NP

PP → (0.47) IN NP |
(0.26) IN S |
(0.09) X Y |
(0.18) IN VP

Number of general rules generated per PCFG grammar	=	4
Number of rules per POS = 5 POS * 3 rules	=	15
Number of rules per option = 24 options * 5 rules	=	120
Number of alternate manifestation rules = n (n+1) /2		
where n = 3 (for S)		6
n = 2 (for VBD)		3
n = 6 (for NP)		21
n = 9 (for VP)		45
n = 4 (for PP)		10
Total GenProb rules	=	219

4.5 Summary of PCFG Encoding

The implicit constraints of PCFG have been identified and described in terms of constraints to be enforced. Furthermore, these constraints have been implemented in the GenProb language.

PCFG is a general formalism that captures regularities in several domains, a behavior we would like from AGI systems. However, PCFGs encode only certain kinds of constraints. By translating PCFGs into a more general probabilistic framework, joint reasoning over PCFG and other constraints is possible. The constraints of PCFG have been identified and encoded in a relational language that in addition to capturing causal conditional probabilities can also represent (potentially cyclic) boolean constraints. An example application of this integration of PCFG and probabilistic relational constraints is in the domain of language understanding. Knowledge of linguistic syntax encoded in PCFG can interact with the generated semantics of the sentence and also the world knowledge encoded in the system to effectively solve problems like lexical (or word sense) ambiguity. In the future, we would like to integrate the constraints of richer grammars like lexicalized grammars (Head- driven Phrase Structure Grammar etc) with this general representation.

5. Contribution

Many difficulties in natural language understanding have been attributed to uncertainty in language and the ability of language grammars to generate an infinite number of sentences. In order to make progress, research in language understanding has progressed on isolated sub problems such as part of speech tagging and shown quantitative accuracies over them. However, in recent years these approaches have shown relatively low improvements and the most promising approaches have been those resolving uncertainties through integrating information from sources beyond those conventionally thought to belong within a sub-problem(Collins 1999; Resnik 1999).

Our approach to language understanding has been to define a formalism that can capture information across multiple sources (vision, auditory, background knowledge, etc.) and perform joint reasoning over the knowledge encoded in this representation. The practical difficulty with this approach is that current reasoning systems are capable of either handling uncertainties in a mathematically well-founded fashion, like the Bayesian networks or handling infinite number of words like CYK parsers. Hence, in order to handle inference required by NLP which involves both an infinite domain and uncertainties, a new inference framework must be employed. Our contribution to the field of NLP can be summarized as follows:

1. *Language for representing both syntax and semantics.* Defining a language that is capable of representing linguistic constraints poses several difficulties, for such a language should have all of the following capabilities: the language must be *relational* i.e. allow generalizations in the form of quantified statements, be *probabilistic* indicating that the result has uncertainty, allow the representation of *unknown objects* like new sentences and have mechanisms that provide *guaranteed inference* over this representation. However, existing probabilistic systems have inference mechanisms that are most efficient when the theories are simple and have no relational representation (quantifications). Introducing a language that demands both probabilistic and relational representation requires the development of a new inference algorithm.

A few relational probabilistic languages have been introduced(Singla and Domingos 2006); however these languages circumvent the demand of a specialized inference algorithm by enforcing restrictions (like domain-closure) on the types of generalizations and internally converting these relational representations into simple theories that can be handled by conventional inference engines. When the relational language is required to handle objects that are not pre-defined before inference, as in the case of linguistic constraints, relational theories can no longer be simplified, thus demanding innovation to the field in terms of inference algorithms. Our first contribution is defining a language, GenProb, capable of representing linguistic constraints by ensuring that GenProb is relational, probabilistic and allows unknown objects.

2. *Mapping linguistic constraints onto the GenProb language.* The next step is to demonstrate that the GenProb language has the expressive power to capture well-studied linguistic grammars. The mapping of linguistic constraints onto non-linguistic theory is difficult because sophisticated linguistic concepts like movement and gaps that explain relative clauses have added layers of complexity to the linguistic theories. Finding parallels of highly structured linguistic grammar concepts in domain-general representations, though explored in the field (CITE Cassimatis 04), has not been done for full-scale grammars.

As a proof of concept that linguistic grammar can be represented in GenProb, we have shown how probabilistic context free grammar can be represented. Our contribution by this mapping of PCFG onto GenProb language is showing that by representing both world knowledge and syntactic theory in the same formalism, the same general inference algorithm, which reasons over world knowledge, can now be used to parse language.

3. *Guaranteed inference over GenProb.* The basis of this thesis' contribution is the requirement that language truly needs relational representation and unknown objects which are beyond theories that assume domain-closure. Hence, tackling inference over such a theory that posits novel objects during inference is a novel contribution to the field. The major challenge of an inference algorithm for a

language like the GenProb language which supports relational representation and allows unknown objects, is that the language licenses a solution that is potentially infinite. Therefore, defining a guaranteed inference algorithm using traditional model finding approaches is not feasible.

Our contribution has been to define two characteristics that identify a sub-class of GenProb problems for which we can provide guaranteed results, and defining an inference algorithm for this sub-class of problems:

a. *Relevant models.* A GenProb theory with constraints like every mother has a mother requires the positing of a mother object for every new mother object created as well. The solution of such a GenProb theory might lead to a model of infinite size, i.e. a solution with an infinite number of objects (in this case mothers) and in-turn an infinite number of grounded (simple non-quantified) constraints. Traditional model-finding algorithms initialize all possible models and search over this space of possible models. However, in the case of theories with unknown objects, which lead to infinite models, it is not possible to initialize and enumerate all models, making traditional model-finding approaches unusable.

A GenProb theory which allows possibly infinite models can at times have a few relevant finite models. Our contribution of defining the theory of relevant models, allows us to examine a finite relevant model as an alternate solution to an infinite model.

b. *Increasing cost models theory.* Even though a GenProb theory might have a set of relevant models associated with it, the goal is to find the solution or model of the least cost. The difficulty in calculating costs of these models is again that some of those models might possibly be infinite, leading to non-trivial calculations.

By defining characteristics of GenProb theory and generalizing over them, we are able to predict the trend of costs even for infinite models. One such class of GenProb theories is defined as increasing cost model theory. An increasing cost model theory has the property that, given that the theory has many solutions, a solution to the theory with infinite number of objects can never be better than a finite model solution.

Our contribution by defining the concepts of increasing cost models theory and relevant models is that we can now guarantee complete inference on a subset of GenProb theory that is an increasing cost model theory and has a finite relevant model. Furthermore, we can show that typical PCFGs (those where the self generating non-terminal rules are not of strength 1) belong to the subset of GenProb theory for which complete inference can be guaranteed.

c. *Algorithm for inference over unknown objects.* Designing an inference algorithm for GenProb is difficult because the algorithm has to dynamically adding objects, keep track of these posited objects and make sure that the algorithm terminates within a practical time-limit.

Our contribution to inference over relational probabilistic representation is to use a complete algorithm, GenDPLL, instead of the more traditional simulation based approaches like MCMC. GenDPLL allows the positing of previously unknown objects and lazily grounds relational constraints on a need basis.

Future work

There are several directions in which this work can be extended to further demonstrate syntax and semantics interaction. A linguistic grammar with built-in support for generative semantics, when encoded in general constraints, would ideally showcase the interaction of syntax and semantics. However, PCFG does not have explicit support for semantics. Encoding a theoretically rich grammar like the Head-driven Phrase Structure Grammar (Sag et al. 2003), though challenging, is an ideal next step.

This research would also benefit from quality measures of evaluating the success of semantics interaction. Automatic translation of large sources of world knowledge like Cyc to GenProb constraints is a possible area of future research. Generating justifications for relational probabilistic reasoning is an interesting field of research.

In summary, this thesis by proposing a language, GenProb, that is capable of representing linguistic constraints (as demonstrated with PCFG) in a generic domain-general fashion, and defining a complete inference algorithm, GenDPLL, for this language, has shown a potential approach in which syntax and semantics (and other information like visual scene & background knowledge) can be integrated to perform NLU.

Glossary

Domain closure – The property of a problem to have a finite number of objects in its domain.

GenProb – (or) Generative Probabilistic Theory - A language that is capable of representing probabilistic constraints over infinite models.

GenSAT – (or) Generative Satisfiability Theory – A language that is capable of representing weighted constraints over infinite models, and also has built-in support for mandatory causation.

GenDPLL – (or) Generative DPLL – An algorithm that performs complete inference over a specified subset (increasing cost theory and finite relevant models) of the problems represented in GenSAT.

Generative domain – A domain (like language) which can generate infinite and novel results (in this case phrases).

HPSG – Head-driven Phrase Structure Grammar - A lexicalized linguistic grammar which also allows the representation of semantics.

Infinite model – An assignment of truth value to a theory or a state of the world where the number of elements that are assigned truth values is infinite.

Increasing cost theory – A theory expressed in a generative cost-based language (e.g. GenSAT), for which the cost of its infinite models is always greater than those of its finite models.

Mandatory causation constraint – A rule that states that an event cannot occur without at least one of its causes being true.

Mandatory manifestation constraint – A rule that states that an event should produce at least one of its possible causes.

Model (of a theory) – An assignment of truth value to all the elements of the theory which may be referred to as the state of the world.

PCFG – Probabilistic Context Free Grammar – A linguistic grammar used to represent the syntactic regularity in language. However, PCFG has been used to capture regularity in a number of fields other than linguistics.

Propositional formalism – (completely grounded formalism) – A language that allows expressions containing propositions and connectives. Quantifications and variables cannot be expressed in this formalism.

Relational formalism – A language which allows quantifiers as those in first-order predicate logic or a limited version of it.

Relevant models – Among the number of valid models available for a single theory, some of the models can be grouped as models relevant to a particular partial solution. It is to be noted that some of these relevant models can be infinite models.

SAT - Satisfiability is the problem of determining if the variables of a given boolean formula can be assigned in such a way as to make the formula evaluate to TRUE.

Unknown objects – In a generative domain, objects unknown to be a valid domain member may be generated. In the specific case of GenProb and GenSAT, object unknown to belong to the domain before inference, but are generated during inference are called unknown objects.

Weighted SAT – A satisfiability problem where each of the constraints has a cost associated with it, and the best solution is one for which the sum of the cost of broken constraints is minimum.

CITATIONS

Allen, J., Ferguson G., and Stent A. (2001). An Architecture for more Realistic Conversational Systems. *In Proceedings of the 6th Conference on Intelligent User Interfaces* : 1--8.

Allen, J., Schubert L., et al. (1994). The TRAINS Project: A case study in building a conversational agent. *Journal of Experimental and Theoretical AI*, 7: 7--48.

Berrada, I. and Ferland J. (1996). Michelon: A Multi-Objective Approach to Nurse Scheduling with both Hard and Soft Constraints, *Socio-Economic Planning Science*, 30**:** 183--193.

Bello, P. and Cassimatis N. L. (2006). Developmental Accounts of Theory-of-Mind Acquisition: Achieving Clarity via Computational Cognitive Modeling. *In Proceedings of 28th Annual Conference of the Cognitive Science Society.*

Bello, P. and Cassimatis N. L. (2007). Some Computational Desiderata for Recognizing and Reasoning About the Intentions of Others. In *Proceedings of AAAI Spring Symposium*. Technical Report SS-07-03. pp. 1-6. Ed. George Ferguson.

Bugajska, M. and Cassimatis N. L. (2006). Beyond Association: Social Cognition in Word Learning. *In Proceedings of the International Conference on Development and Learning*.

Burke, E. K., De Causmaecker P., et al. (2004). The State of the Art of Nurse Rostering. *Journal of Scheduling*, 7: 441--499.

Carroll, J., Briscoe E. J., et al. (1998). Parser evaluation: a survey and a new proposal. *In Proceedings of the 1st International Conference on Lexical Resources and Evaluation***:** 447--454.

Cassimatis, N. L. (2005). Integrating Cognitive Models Based on Different Computational Methods. *In Proceedings of Twenty-Seventh Annual Conference of the Cognitive Science Society*.

Cassimatis, N. L. (2006). A Cognitive Substrate for Human-Level Intelligence. *Artificial Intelligence Magazine*, vol. 27.

Cassimatis, N. L. (2008). Resolving Ambiguous, Implicit and Non-Literal References by Jointly Reasoning over Linguistic and Non-Linguistic Knowledge. *In the Proceedings of 12th Annual SEMDIAL Workshop on Semantics and Pragmatics - LONDIAL 2008*.

Cassimatis, N. L., Murugesan A. and Bignoli P. (2009). Inference with Relational Theories over Infinite Domains. *In Proceedings of FLAIRS.*

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. *In Proceedings of the Fourteenth National Conference on Artificial Intelligence, Menlo Park*. AAAI Press/MIT Press.

Charniak, E. (2000). A Maximum-Entropy-Inspired Parser. *Proceedings of 1st North American chapter of the Association for Computational Linguistics*. ACM International Conference Proceeding Series, 4: 132--139.

Cheang, B., Li H., et al. (2003). Nurse Rostering Problems - a Bibliographic Survey. *European Journal of Operational Research*, 151: 447--460.

Collins, M. (1996). A New Statistical Parser Based on Bigram Lexical Dependencies. *In the Proceedings of the 34th annual meeting on Association for Computational Linguistics,* 184--191, Santa Cruz, California.

Collins, M. (2003). Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics,* MIT Press, 29: 589--637.

Croft, W. and Cruse A. (2004). Cognitive Linguistics, *Cambridge University Press*.

Davis, M., Logemann G., et al. (1962). A Machine Program for Theorem Proving. *Communications of the ACM*, 5.

Davis, M. and Putnam H. (1960). A Computing Procedure for Quantification Theory. *Journal of the ACM*, 7: 201--215.

Dechter, R. (2003). Constraint Processing, *Morgan Kaufmann*.

Domingos, P. and Richardson M. (2004). Markov Logic: A Unifying Framework for Statistical Relational Learning. *In Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields***:** 49--54.

Ferguson, G. and Allen J. (1998). TRIPS: An Intelligent Integrated Problem-Solving Assistant. *In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)***:** 567--573.

Frazier, L. and Rayner K. (1987). Resolution of Syntactic Category Ambiguities: Eye Movements in Parsing Lexically Ambiguous Sentences. *Journal of Memory and Language*, 26: 505-526.

Hall, P. and McKevitt P. (1995). Integrating Vision Processing and Natural Language Processing with a Clinical Application. *In the Proceedings of the 2nd New Zealand Two-Stream International Conference on Artificial Neural Networks and Expert Systems (ANNES '95)*

Hindle, D. and Rooth M. (1993). Structural ambiguity and lexical relations. *Computational Linguistics*, 19: 103--120.

Hirsch, H. G. and Pearce D. (2000). Aurora Experimental Framework for the Performance Evaluation of Speech Recognition Systems under Noisy Conditions. *ISCA ITRW ASR2000 (Automatic Speech Recognition: Challenges for the Next Millennium.*

Jackendoff, R. (1983). Semantics and Cognition, *MIT Press*.

Jackendoff, R. (1990). Semantic Structures, *MIT Press*.

Jackendoff, R. (1997). The Architecture of the Language Faculty, *MIT Press*.

Jackendoff, R. (2002). Foundations of Language, *University Press*.

Johnson-Laird, P. N. (1983). Mental Models, *University Press*.

Josef, M. and Robert D. (2002). Using the WordNet Hierarchy for Associative Anaphora Resolution. *COLING-02 on SEMANET: building and using semantic networks - Volume 11, Association for Computational Linguistics***,** 1-7.

Klein, D. and Manning C. D. (2003). Accurate Unlexicalized Parsing. *In Proceedings of the 41st Meeting of the Association for Computational Linguistics*.

Klein, D. and Manning C. D. (2003). Fast Exact Inference with a Factored Model for Natural Language Parsing. *In Advances in Neural Information Processing Systems*, MIT Press**:** 3--10.

Langacker, R. (1987). Foundations of Cognitive Grammar, *Stanford University Press.*

Loveland, D. W. (1968). Mechanical Theorem Proving by Model Elimination. *Journal of Association for Computing Machinery*, 15: 236--251.

McShane, M. (2008). Achieving Human-level NLP: The Case of Reference Resolution. *In Preparation*.

Milch, B., Marthi B., et al. (2004). Blog: Relational Modeling with Unknown Objects. *In ICML 2004 Workshop on Statistical Relational Learning and Its Connections to Other Fields*.

Milch, B., Marthi B., et al. (2007). Blog: Probabilistic models with unknown objects. *In Introduction to Statistical Relational Learning*.

Minsky, M. (1974). A Framework for Representing Knowledge. *MIT-AI Lab Memo*, MIT Press.

Montazeri, N., Ghassem-Sani G., et al. (2006). A Fast and Robust Parser Based on Viterbi Algorithm. *Proceedings of the 11th Annual Int. CSI Computer Conference (CSICC'2006)*, II : 473-478.

Murugesan A. and Cassimatis N. L. (2006). A Model of Syntactic Parsing Based on Domain-General Cognitive Mechanisms. *In Proceedings of 28th Annual Conference of the Cognitive Science Society.*

Pinker, S. and Jackendoff R. (2005). The Faculty of Language: What's Special about it? *Cognition*, 95**:** 201--236.

Resnik, P. (1999). Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal Artificial Intelligence Research*.

Richardson, M. and Domingos P. (2006). Markov Logic Networks. *Machine Learning***:** 107--136.

Sag I., Wasow T. and Bender E. (2003). *Syntactic Theory: A Formal Introduction.* CSLI Lecture Notes No. 152.

Sang, T., Beame P., et al. (2005). Solving Bayesian Networks by Weighted Model Counting. *In Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, AAAI Press**:** 475--482.

Selkirk, E. O. (1984). Phonology and Syntax: The Relation between Sound and Structure, *MIT Press*.

Selkirk, E. O. (1995). Sentence Prosody: Intonation. *The Handbook of Phonological Theory***:** 550--569.

Simone Paolo, P. and Michael S. (2006). Exploiting Semantic Role Labeling, WordNet and Wikipedia for Coreference Resolution. *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. New York, New York, Association for Computational Linguistics**:** 192-199.

Singla, P. and Domingos P. (2006). Memory-efficient inference in relational domains. *Proceedings of the Twenty-First National Conference on Artificial Intelligence*.

Talmy, L. (1975). Semantics and syntax of motion. *In J. P. Kimball (Ed.), Syntax and Semantics*, 4: 181--238.

Talmy, L. (1985). Lexicalization Patterns: Semantic Structure In Lexical Forms. *Language Typology And Syntactic Description. Vol. 3, Grammatical Categories and The Lexicon*, University Press.

Talmy, L. (1988). Force Dynamics in Language and Cognition. *Cognitive Science*, 2: 49--100.

Wolf, A. (1998). P-SETHEO: Strategy Parallelism in Automated Theorem Proving. *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX-98)*, Springer, LNAI**:** 32--0.

Yarowsky, D. (1992). Word Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora. *In the 14th International Conference on Computational Linguistics, Nates*.