Parsing PCFG within a General Probabilistic Inference Framework

Arthi Murugesan
Department of Cognitive Science
Rensselaer Polytechnic Institute
Troy NY 12180

Nicholas L. Cassimatis
Department of Cognitive Science
Rensselaer Polytechnic Institute
Troy NY 12180

Abstract

One of the aims of Artificial General Intelligence(AGI) is to use the same methods to reason over a large number of problems spanning different domains. Therefore, advancing general tools that are used in a number of domains like language, vision and intention reading is a step toward AGI. Probabilistic Context Free Grammar (PCFG) is one such formalism used in many domains. However, many of these problems can be dealt with more effectively if relationships beyond those encoded in PCFGs (category, order and parthood) can be included in inference. One obstacle to using more general inference approaches for PCFG parsing is that these approaches often require all state variables in a domain to be known in advance. However, since some PCFGs license infinite derivations, it is in general impossible to know all state variables before inference. Here, we show how to express PCFGs in a new probabilistic framework that enables inference over unknown objects. This approach enables joint reasoning over both constraints encoded by a PCFG and other constraints relevant to a problem. These constraints can be encoded in a first-order language that in addition to encoding causal conditional probabilities can also represent (potentially cyclic) boolean constraints.

Introduction

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).

Probabilistic Context Free Grammars (or Stochastic Context Free Grammars) encode a few types of relations like temporal ordering, category and parthood. These kinds of relations play an important role in a wide variety of domains, including natural language (Charniak, 2000), the secondary structure of RNA (Sakakibara, Brown, Underwood, Mian & Haussler, 1994), computer vision (Moore & Essa, 2002), plan recognition (Pynadath & Wellman, 2000), intention reading and high-level behavior recognition (Nguyen, Bui, Venkatesh & West, 2003).

Though these relations encoded by PCFG can be used in different domains, many problems require the representation of additional relations. Constraints such as causality can not be expressed within PCFG. In the domain of natural language processing, for example, syntactic regularities are captured by the grammar and improvements are obtained by adding more constraints including lexicalization (Collins, 2003). However, visual cues, social context, individual bias and semantics are all factors affecting language processing (Ferguson & Allen, 2007) that have no straightforward PCFG representation.

The additional relations can be represented in more general frameworks such as Bayesian networks and weighted constraint SAT solvers. These systems, besides modeling PCFG constraints, can also encode a variety of other constraints within the same framework. However, these systems typically require all objects or state variables in a domain to be known in advance and thus are poorly suited for PCFG parsing, which can lead to infinite derivations.

A few probabilistic inference approaches deal with problems that have a large number of grounded constraints by utilizing on demand or lazy grounding of constraints (Domingos et al. 2006). However, these systems nevertheless require that all the possible objects of the domain be declared in advance.

Approach

Here, we describe a new general probabilistic inference framework that allows inference over objects that are not known in advance, but instead are generated as needed. This key feature makes it possible to harness the power of PCFGs and the full flexibility of more general frameworks within a single, integrated system. Our approach to encoding and parsing PCFG in a general probabilistic inference framework has three features:

Explicitly Encode Constraints Implicit in PCFG

Implicit in PCFG are several constraints. For example, (a) every nonterminal in a derivation must ultimately be manifest as a terminal and (b) every phrase can be immediately dominated by only one other phrase. Explicitly encoding these constraints in a more expressive probabilistic inference framework allows them to be jointly reasoned over with other forms of constraints.

Relational Representation

Even with grammars that license only finite derivations, the number of such derivations can be very large. This translates into inference problems with large numbers of state variables, and the resulting memory demands can render this kind of problem intractable. One way to overcome this is to use relational probabilistic languages, for which there are inference approaches that significantly reduce the memory demands imposed by large numbers of state variables (Domingos, Kok, Poon, Richardson & Singla, 2006).

Licensing the Existence of Unknown Objects

We will use a probabilistic framework, GenProb, that enables reasoning over objects not known prior to inference.

Generative Probabilistic Theory

These three desiderata mentioned are manifest in the Generative Probabilistic theory, GenProb (under review). GenProb is a relational language for expressing probabilistic constraints over unknown objects. This language supports both causal conditional probabilities and (potentially cyclic) boolean constraints. An exact inference algorithm has been defined for GenProb theories (under review) that can be classified as increasing cost models. PCFG problems are increasing cost models and hence exact reasoning over these possibly infinite models is possible.

Syntax of GenProb

GenProb is a language for expressing probabilistic relational theories over unknown objects. 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 .01% of the time. Planes being tracked are fitted with collision warning systems that, in the presence of other planes in range, have a 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()` $\longrightarrow$(.01) `InRange(?p)` $\wedge$ `Plane(?p)`
`True()` $\longrightarrow$(.001) `WeakBlip(?b)`
The causal aspects of the theory are indicated with conditionals:
`InRange(?p)` $\wedge$ `Plane(?p)` $\longrightarrow$
`(.3)StrongBlip(?b), (.5)WeakBlip(?b) ,`
`(.2) NoBlip(?b), ?p`
`Detect(?p1, ?p2)` $\wedge$ `Plane(?p1)` $\wedge$ `Plane(?p2)` $\wedge$
`InRange` $\longrightarrow$ `(.9)TransitAlarm(?p1), ?p1, ?p2`

The first conditional indicates that a plane that is in range will have one and only one of the following effects: a strong blip (in 30% of cases), an uncertain blip (50%), and no blip otherwise. The occurrence of ?p 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:
`TransitAlarm(?p1)` $\Longrightarrow$ `AlarmSound()`
Conditionals with numbers are called causal conditionals and those without numbers are called logical conditionals. Logical conditionals are hard constraints.

Since blips occur in the consequents of causal conditionals, they must be the effect of one of these conditionals. In this case, strong blips can only be caused by planes, while weak blips can be caused by planes and by noise. We label the causal interpretation that an effect must be caused by one of its causes(constraint's antecedents being true) as *mandatory causation.* Such mandatory causation is not implied by logical conditionals. Mandatory causation for literals can be neutralized with a causal conditional whose antecedent is True(), which (see below) is always true.

More formally, a GenProb theory is a set of causal and logical conditionals. Causal conditionals are of the form $C_1 \wedge ... \wedge C_n \longrightarrow (p_1)E_1, ..., (p_m)E_m, ...?v_i, ...$, 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 are conjunctions of literals. Each E_i conjunction is called an *effect* of the conditional 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 conditionals are of the form $A_1 \wedge ... \wedge A_n \Longrightarrow B_1 \wedge ... \wedge B_n$,where each conjunct is either a literal or a negation thereof. Literal a is a grounding of literal b if they are equivalent under an assignment of variables to objects in b and no variables occur in a. Literals and conditionals are grounded if they contain no variables.

Exact Inference Over GenProb

Many of the existing approaches for combining first-order logic and probabilistic graphical models propositionalize relational theories and making inferences over these propositionalized formulae. However, most of these approaches require all objects in the domain to be known in advance, although many important problems like probabilistic context-free grammars involve objects that are initially unknown and permit infinite derivations.

Theories over potentially unknown objects pose two problems for inference approaches based on propositionalization. First, theories of finite size that express relations over unknown objects often require infinite models. For example, the formula, $Mammal(a) \wedge \forall x(Mammal(x) \longrightarrow Mammal(mother(x)))$ (together with formulae stating that a mammal cannot be its own ancestor) require an infinite model because as mother must also have a mother who must also have a mother, ad infinitum. Likewise, some context-free grammars with finite numbers of rules and terminals can generate an infinite number of sentences. Since an algorithm cannot enumerate an infinite model in finite time, we must find a way of finitely characterizing solutions to problems that have infinite models.

A second problem associated with unknown objects is that even if all models of a theory can be finitely characterized, there may nevertheless be infinitely many such models. Complete satisfiability algorithms (e.g., those based on Davis-Putnam-Logemann-Lovelan DPLL algorithm) 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. Exact inference over a subset of GenProb theories has been defined (under review).

The key approach behind the inference mechanism, is to convert GenProb theory to a corresponding weighted satisfiability (SAT) model. However, since GenProb licenses unknown objects, this weighted SAT model must also allow the licensing of unknown objects during inference. Therefore, a version of SAT called the Generative SAT(GenSAT) has been defined. Also an exact inference algorithm, Generative DPLL (GenDPLL), that makes guaranteed inference over GenSAT constraints is defined. GenDPLL is a DPLL-like branch-and-bound algorithm that lazily posits new objects and instantiates clauses involving them. It has been prooved that GenDPLL is guaranteed to find finite relevant models of certain classes of GenSAT theories with infinite models, which we call increasing cost models.

Increasing cost models are theories in which the introduction of new constraints can only lead to models of lower cost. PCFG is one such theory, because the introduction of more branches to a parse tree always leads to a less probable solution (or an increased cost model).

Mapping PCFG onto GenProb Language

Jointly reasoning over PCFG and other constraints is enabled by expressing PCFG problems in the GenProb language and using the defined inference mechanisms of GenProb to reason over these constraints. A PCFG rule is of the form:
$X \rightarrow (Prob_1)u_{11}u_{12} \ldots u_{1m_1}$
$\mid (Prob_2)u_{21}u_{22} \ldots u_{2m_2}$

...
$\mid (Prob_n)u_{n1}u_{n2} \ldots u_{nm_n}$

where the antecedent X on the 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 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 probability of the non-terminal generation a particular set of symbols like $u_{11}u_{1m_1}$ or $u_{21}u_{2m_2}$ 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.

The Order Constraint

Probabilistic rules 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 depends on several ordered components including the order of the list of terminals in the string (I) and the order of right hand side(RHS) components in a PCFG rule.

Ordering of Input

Let the input I, say A1, A2, An, 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 1. capturing the time of occurrence of an event(both start and end points) and 2. establishing relations between these time of occurrences. The constraints of I (A1, A2 .. An) is captured using the following grounded propositions.

```
Occur(a1), IsA(a1, A1),
StartTime(a1, t1), EndTime(a1, t1),
ImmediatelyBefore(t1, t2) ,
Occur(a2), IsA(a2, A2),
StartTime(a2, t2), EndTime(a2, t2),
ImmediatelyBefore(t2, t3) ,
...
Occur(an-1), IsA(an-1, An-1),
StartTime(an-1, tn-1), EndTime(an-1, tn-1),
ImmediatelyBefore(tn-1, tn),
Occur(an), IsA(an, An),
StartTime(an, tn), EndTime(an, tn)
```

Order of PCFG Rule Arguments

A typical PCFG rule(R) of format $X \rightarrow (Prob)u_1u_2$ ... u_m depends on the order of the u symbols. According to the definition of R, u symbols can be both terminals and 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. (We'll also expand on this in the creation of new phrases section)

```
Occur(?xobj) ∧ IsA(?xobj, X) ∧
StartTime(?xobj, ?t0) ∧ EndTime(?xobj, ?tn)
⟶ (Prob)
Occur(?u1obj) ∧ IsA(?u1obj, u1) ∧
StartTime(?u1obj,?t0) ∧ EndTime(?u1obj, ?t1)
∧ ImmediatelyBefore ( ?t1, ?t2) ∧
...
Occur(?unobj) ∧ IsA(?unobj, un) ∧
StartTime(?unobj, ?t(n-1)) ∧
EndTime(?unobj, ?tn)
```

Creation of New Objects

The GenProb constraints that capture the PCFG generation rules have unbound objects on both sides as shown in the ordering constraint of R . The GenProb language handles an unbound constraint by creating a new object for the unbound variable in the LHS when the pattern in the RHS is matched completely. The new object is created through the process of skolemization.

Specifically with respect to the rule of the ordering constraint, when the objects of the RHS and their corresponding category, time information match the pattern, the ?xObj on the LHS is created. Also the time information which is already bound by the RHS pattern matching, is asserted for the new object.

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.

The 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, ?parentNode1) ∧
PartOf( ?childNode, ?parentNode2) ∧
NOT Same(?parentNode1, ?parentNode2)
⟹ FALSE
```

Mandatory Parent

The GenProb language handles two kinds of constraints: causal and logical constraints. Any consequent of a causal constraint is required (according to mandatory causation) to have at least one of its causes to be true. Thus, if $P_1(a) \longrightarrow R(a), P_2(a) \longrightarrow R(a)$, ... , $P_n(a) \longrightarrow R(a)$ are all the causal conditionals with R(a) in the consequent, any model where R(a) is true must have at least one of $P_i(a)$ being true. This is captured in GenProb by keeping track of all the causes of grounded propositions and adding a logical constraints called the mandatory causation constraint.

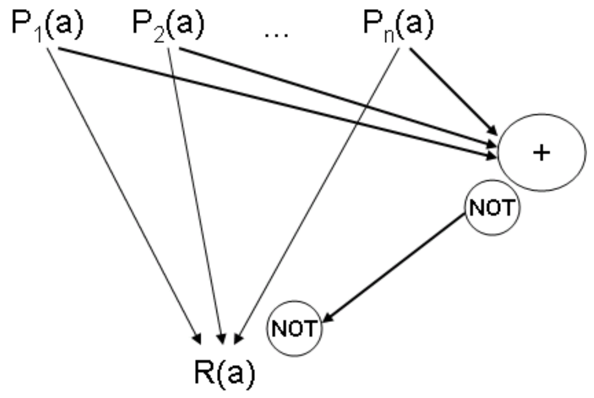

Figure 1: Captures the mandatory parent rule that the node R(a) cannot exist with at least one of its parents: NOT $P_1(a) \wedge NOT P_2(a) \wedge ... \wedge NOT P_n(a) \Longrightarrow$ NOT R(a)

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 captured in GenProb language. Hence there can be no node that is unconnected to the parse tree.

Unique Manifestation

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

$X \rightarrow (Pr_1)f_1f_2$ ... f_m
$| \ (Pr_2)s_1s_2$ ... s_m

A particular non-terminal node can only generate one of the options. This implicit constraint of unique representation among alternate options is captured using the comma (,) symbol in the GenProb language and listing the mutually exclusive options with their probabilities in the same GenProb constraint. The internal weighted constraint representation of a grounded GenProb constraint of this format is shown in Figure 2.

Start Symbol

The one node in the parse tree that does not have a parent is the start node S. This constraint is captured in the GenProb language by assigning a high prior value to the object with the category of the start symbol and its time of occurrence spanning over the entire length of the input string I.

```
TRUE ⟹
IsA(?obj, S) ∧ Occur(?obj) ∧
```

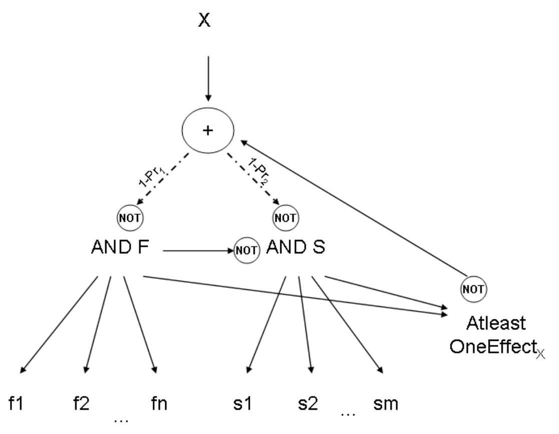

Figure 2: Shows the underlying weighted constraints of $X \rightarrow (Pr_1) f_1 f_2 \,...\, f_m \mid (Pr_2) s_1 s_2 \,...\, s_m$

`StartTime(?obj, Istrt)` $\wedge$ `EndTime(?obj, Iend)`

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 generates 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.

This constraint of saying that among all the possible generations of a non-terminal at least one of them should hold true is harder to capture. We have introduced a corresponding AtleastOneEffect proposition for every non-terminal node(Figure 1). The only causes for the AtleastOneEffect proposition of a non-terminal are the RHS components of the productions in R for this particular non-terminal. Since GenProb language has the built in causal tendency to falsify an event when all its causes are false, the only reason for AtleastOneEffect proposition to be true is if one of the productions in R, the rule set of PCFG, has been generated.

`Occur(?obj)` $\wedge$ `NOT AtleastOneEffect(?obj)`
$\Longrightarrow$ `FALSE`

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

$X \rightarrow (0.75) a \mid (0.25) YZ$

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

1.
`Occur(?aobj)` $\wedge$ `IsA(?aobj, a)` $\wedge$
`StartTime(?aobj,?tStart)` $\wedge$
`EndTime(?aobj, ?tEnd)` $\wedge$
`Occur(?xobj)` $\wedge$ `IsA(?xobj, X)` $\wedge$
`StartTime(?xobj,?tStart)` $\wedge$
`EndTime(?xobj, ?tEnd)`
$\Longrightarrow$ `AtleastOneEffect(?xobj)`

2.
`Occur(?yobj)` $\wedge$ `IsA(?yobj, Y)` $\wedge$
`StartTime(?yobj,?tStart)` $\wedge$
`EndTime(?yobj, ?tMid1)` $\wedge$
`ImmediatelyBefore(?tMid1, ?tMid2)` $\wedge$
`Occur(?zobj)` $\wedge$ `IsA(?zobj, Z)` $\wedge$
`StartTime(?zobj,?tMid2)` $\wedge$
`EndTime(?zobj, ?tEnd)` $\wedge$
`Occur(?xobj)` $\wedge$ `IsA(?xobj, X)` $\wedge$
`StartTime(?xobj,?tStart)` $\wedge$
`EndTime(?xobj, ?tEnd)`
$\Longrightarrow$ `AtleastOneEffect(?xobj)`

Though the mandatory manifestation constraint ensures that there is no unexpanded non-terminal in the tree, it is not guaranteed for the parse tree to end in terminals. PCFG rules of the form X1 → X11 and X11 → X1 can lead to an endless depth of the parse tree.

An Example of Interaction Enabled By GenProb

The importance of representing syntactic grammar in the same general formalism that also allows relational representations and causal conditionals is that syntax can now interact with other aspects of language like semantics, background knowledge and visual perception. For example, problems like part-of-speech tagging and word sense disambiguation, which are conventionally studied as isolated sub-problems, can be addressed by this interaction of syntax and semantics.

In order to demonstrate an example, let us consider the word "bug". According to Wordnet, the word "bug" has the coarse senses of the nouns insect animal, system error and listening device, and also the verbs annoy and eavesdrop in this order of frequency. Given we have the corresponding frequency of these senses(Probi), the following constraint can be added to the system:

`IsA(?word, Word)` $\wedge$ `HasPhonology(?word, soundBug)`
$\longrightarrow$
`(Prob1) HasWordSense(?word, animalBug),`
`(Prob2) HasWordSense(?word, systemErrorBug),`
`(Prob3) HasWordSense(?word, deviceBug),`
`(Prob4) HasWordSense(?word, annoyVerbBug),`
`(Prob5) HasWordSense(?word, evesdropVerbBug)`

By default with no further information the most frequent sense of the word is preferred. However, consider the example sentence "The bug needs a battery". In this case, the bug refers to the noun listening device because animals and abstract concepts like errors do not require batteries, which say is available background knowledge. As the sentence is parsed and semantics is generated within the same framework, the generated semantics that an animal needs battery or that an abstract entity needs battery creates contradiction with the background knowledge. Hence, the inference system with this information concludes that the correct interpretation of the word bug is the listening device. As an illustration, we show how the required background knowledge can be represented in GenProb.

```
IsA(?obj, Organic) ⟹ IsA(?obj, Physical)
IsA(?obj, Inorganic) ⟹ IsA(?obj, Physical)
IsA(?obj, Physical) ⟹ IsA(?obj, Entity)
IsA(?obj, Abstract) ⟹ IsA(?obj, Entity)
IsA(?obj, Abstract)
⟹ NOT IsA(?obj, Physical)
IsA(?obj, Organic)
⟹ NOT IsA(?obj, Inorganic)
NOT(?obj, Inorganic)
⟹ NOT Need(?obj, battery)
```

Related Work

Logics over infinite domains have been characterized (Milch et al., 2005, Singla & Domingos, 2007), but to our knowledge no guaranteed inference algorithm for these problems has thus far been published. Several approaches try to generalize PCFG. Hierarchical dirchilet process (Liang, Petrov et.all 2007) represent infinite number of constraints. However, the present approach is the only one to our knowledge that allows exact inference (under review) and combines in logical constraints which need not adhere to cyclicity conditions. Finally, it is anticipated that jointly reasoning over syntactic and semantic constraints in natural language processing applications will require the kind of relational language offered by the present approach.

Conclusion

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.

References

Anonymous. (under review). Inference with Relational Theories over Infinite Domains.

Cassimatis N.L. (2006). A Cognitive Substrate for Human-Level Intelligence. *AI Magazine.* Volume 27 Number 2.

Charniak, E. (2000) A maximum-entropy-inspired parser. *In: Proc. NAACL.* 132-139

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

Moore, D. and Essa, I. (2002) Recognizing multitasked activities from video using stochastic context-free grammar. *In Proceedings of AAAI-02.*

Domingos, P., Kok, S., Poon, H., Richardson, M., and Singla, P. (2006) Unifying Logical and Statistical AI. *Paper presented at the AAAI-06.*

Singla, P., and Domingos, P. (2007) Markov Logic in Infinite Domains. *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (pp. 368-375). Vancouver, Canada: AUAI Press.*

Milch, B., Marthi, B., Russell, S., Sontag, D., Ong, D. L., and Kolobov, A., 2005. "Blog: Probabilistic Models With Unknown Objects." *"Proceedings of the Nineteenth Joint Conference on Artificial Intelligence."*

Ferguson, G., and J. Allen (2007). Mixed-Initiative Dialogue Systems for Collaborative Problem-Solving. *AI Magazine 28(2):23-32. Special Issue on Mixed-Initiative Assistants. AAAI Press.*

Liang, P., Petrov, S., Jordan, M. I., and Klein, D. (2007). The infinite PCFG using hierarchical Dirichlet processes. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Nam T. Nguyen, Hung H. Bui, Svetha Venkatesh, and Geoff West. (2003) Recognising and monitoring highlevel behaviours in complex spatial environments. *In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR-03)*

Pynadath, David V.; Wellman, Michael P. (2000) Probabilistic state-dependent grammars for plan recognition. *In Proceedings of the conference on uncertainty in artificial intelligence* pp. 507-514

Percy Liang Slav Petrov Michael I. Jordan Dan Klein The Infinite PCFG using Hierarchical Dirichlet. (2007) *Processes Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning,* pp. 688-697, Prague, June 2007. c2007 Association for Computational Linguistics

Y. Sakakibara, M. Brown, R. C. Underwood, I. S. Mian, and D. Haussler, "Stochastic context-free grammars for modeling RNA," (2004) *Proceedings of the 27th Annual Hawaii International Conference on System Sciences. Volume 5 : Biotechnology Computing, L. Hunter, Ed. Los Alamitos, CA, USA: IEEE Computer Society Press*, pp. 284-294.