Using model-theoretic semantic interpretation to guide statistical parsing and word recognition in a spoken language interface*

William Schuler
Department of Computer and Information Science
University of Pennsylvania
200 S. 33rd Street
Philadelphia, PA 19104
schuler@linc.cis.upenn.edu

Abstract

This paper describes an extension of the semantic grammars used in conventional statistical spoken language interfaces to allow the probabilities of derived analyses to be conditioned on the meanings or *denotations* of input utterances in the context of an interface's underlying application environment or *world model*. Since these denotations will be used to guide disambiguation in interactive applications, they must be efficiently shared among the many possible analyses that may be assigned to an input utterance. This paper therefore presents a formal restriction on the scope of variables in a semantic grammar which guarantees that the denotations of all possible analyses of an input utterance can be calculated in polynomial time, without undue constraints on the expressivity of the derived semantics. Empirical tests show that this model-theoretic interpretation yields a statistically significant improvement on standard measures of parsing accuracy over a baseline grammar not conditioned on denotations.

1 Introduction

The development of speaker-independent mixed-initiative speech interfaces, in which users not only answer questions but also ask questions and give instructions, is currently limited by the performance of language models based largely on word co-occurrences. Even under ideal circumstances, with large application-specific corpora on which to train, conventional language models are not sufficiently predictive to correctly analyze a wide variety of inputs from a wide variety of speakers, such as might be encountered in a general-purpose interface for directing robots, office assistants, or other agents with complex capabilities. Such tasks may involve unlabeled objects that must be precisely described, and a wider range of actions than a standard database interface would require (which also must be precisely described), introducing a great deal of ambiguity into input processing.

This paper therefore explores the use of a statistical model of language conditioned on the meanings or *denotations* of input utterances in the context of an interface's underlying application environment or *world model*. This use of model-theoretic interpretation represents an important extension to the 'semantic grammars' used in existing statistical spoken language interfaces, which rely on co-occurrences among lexically-determined semantic classes and slot fillers (Miller et al., 1996), in that the probability of an analysis is now also conditioned on the existence of denoted entities and relations in the world model. The advantage of the interpretation-based disambiguation advanced here is that the probability of generating, for example, the noun phrase 'the lemon next to the safe' can be more reliably estimated from the frequency with which noun phrases have non-empty denotations – given the fact that 'the lemon next to the safe' does indeed denote something in the world model – than it can from the relatively sparse co-occurrences of frame labels such as LEMON and NEXT-TO, or of NEXT-TO and SAFE.

Since there are exponentially many word strings attributable to any utterance, and an exponential (Catalan-order) number of possible parse tree analyses attributable to any string of words, this use of model-theoretic interpretation for disambiguation must involve some kind of sharing of partial results between competing analyses if interpretation is to be

*The author would like to thank David Chiang, Karin Kipper, and three anonymous reviewers for particularly helpful comments on this material. This work was supported by NSF grant EIA 0224417.

performed on large numbers of possible analyses in a practical interactive application. This paper therefore also presents a formal restriction on the scope of variables in a semantic grammar (without untoward constraints on the expressivity of the derived semantics) which guarantees that the denotations of all possible analyses of an input utterance can be calculated in polynomial time. Empirical tests show that this use of model-theoretic interpretation in disambiguation yields a statistically significant improvement on standard measures of parsing accuracy over a baseline grammar not conditioned on denotations.

2 Model-theoretic interpretation

In order to determine whether a user's directions denote entities and relations that exist in the world model – and of course, in order to execute those directions once they are disambiguated – it is necessary to precisely represent the meanings of input utterances.

Semantic grammars of the sort employed in current spoken language interfaces for flight reservation tasks (Miller et al., 1996; Seneff et al., 1998) associate fragments of logical – typically relational algebra – expressions with recursive transition networks encoding lexicalized rules in a context-free grammar (the independent probabilities of these rules can then be estimated from a training corpus and multiplied together to give a probability for any given analysis). In flight reservation systems, these associated semantic expressions usually designate entities through a fixed set of constant symbols used as proper names (e.g. for cities and numbered flights); but in applications with unlabeled (perhaps visually-represented) environments, entities must be described by predicating one or more modifiers over some variable, narrowing the set of potential referents by specifying colors, spatial locations, etc., until only the desired entity or entities remain. A semantic grammar for interacting with this kind of unlabeled environment might contain the following rules, using variables $x_1 ... x_n$ (over entities in the world model) in the associated logical expressions:

$$
\begin{array}{ll}
\text{VP} \rightarrow \text{VP PP} & : \lambda x_1 ... x_n .\, \$1(x_1 ... x_m) \wedge \\
 & \quad \$2(x_1, x_{m+1} ... x_n) \\
\text{VP} \rightarrow \text{hold NP} & : \lambda x_1 .\, \mathit{Hold}(\mathit{Agent}, x_1) \wedge \$2(x_1) \\
\text{NP} \rightarrow \text{a glass} & : \lambda x_1 .\, \mathit{Glass}(x_1) \\
\text{PP} \rightarrow \text{under NP} & : \lambda x_1 x_2 .\, \mathit{Under}(x_1, x_2) \wedge \$2(x_2) \\
\text{NP} \rightarrow \text{the faucet} & : \lambda x_1 .\, \mathit{Faucet}(x_1)
\end{array}
$$

in which m and n are integers and $0 \leq m \leq n$. Each lambda expression $\lambda x_1 ... x_n .\, \phi$ indicates a function from a tuple of entities $\langle e_1 ... e_n \rangle$ to a truth value defined by the remainder of the expression ϕ (substituting $e_1 ... e_n$ for $x_1 ... x_n$), which denotes a set of tuples satisfying ϕ, drawn from $\mathcal{E}^n$ (where $\mathcal{E}$ is the set of entities in the world model).

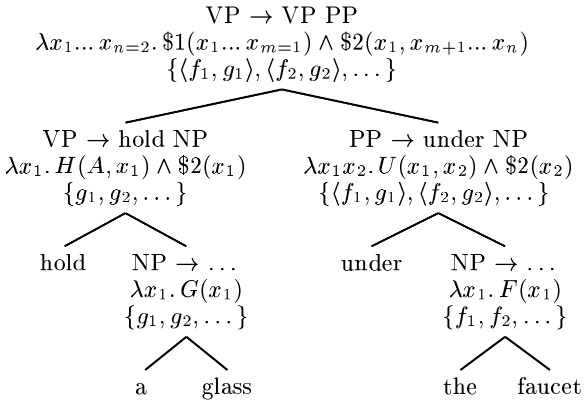

Figure 1: Semantic grammar derivation showing the associated semantics and denotation of each constituent. Entire rules are shown at each step in the derivation in order to make the semantic associations explicit.

The pseudo-variables $\$1, \$2, \ldots$ in this notation indicate the sites at which the semantic expressions associated with each rule's nonterminal symbols are to compose (the numbers correspond to the relative positions of the symbols on the right hand side of each rule, numbered from left to right). Semantic expressions for complete sentences are then formed by composing the sub-expressions associated with each rule at the appropriate sites.[1]

Figure 1 shows the above rules assembled in a derivation of the sentence 'hold a glass under the faucet.' The denotation annotated beneath each constituent is simply the set of variable assignments (for each free variable) that satisfy the constituent's semantics. These denotations exactly capture the meaning (in a given world model) of the assembled semantic expressions dominated by each constituent, regardless of how many sub-expressions are subsumed by that constituent, and can therefore be shared among competing analyses in lieu of the semantic expression itself, as a partial result in model-theoretic interpretation.

2.1 Variable scope

Note, however, that the adjunction of the prepositional phrase modifier 'under the faucet' adds another free variable (x_2) to the semantics of the verb

[1] This use of pseudo-variables is intended to resemble that of the unix program 'yacc,' which has a similar purpose (associating syntax with semantics in constructing compilers for programming languages).

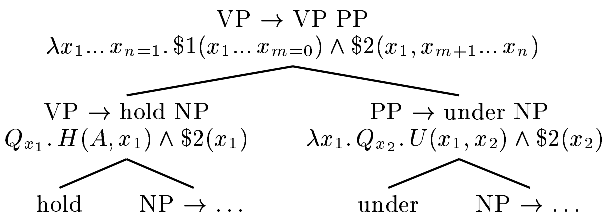

Figure 2: Derivation with minimal scoping. The variable x_1 in the semantic expression associated with the prepositional phrase 'under the faucet' cannot be identified with the variable in the verb phrase.

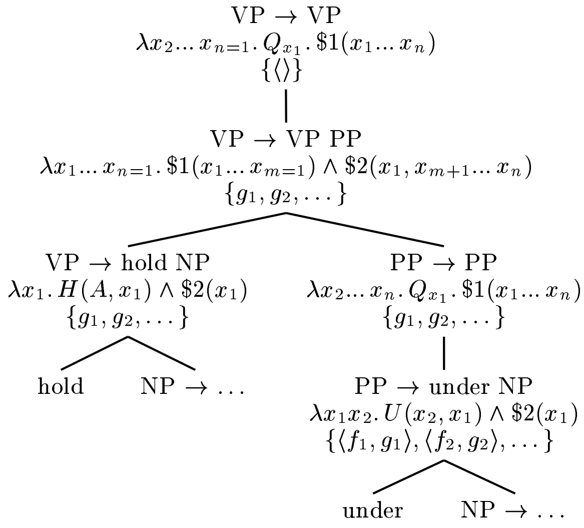

Figure 3: Derivation with desired scoping.

phrase, and therefore another factor of $|\mathcal{E}|$ to the cardinality of its denotation. Moreover, under this kind of global scoping, if additional prepositional phrases are adjoined, they would each contribute yet another free variable, increasing the complexity of the denotation by an additional factor of $|\mathcal{E}|$, making the shared interpretation of such structures potentially exponential on the length of the input. This proliferation of free variables means that the variables introduced by the noun phrases in an utterance, such as 'hold a glass under the faucet,' cannot all be given global scope, as in Figure 1. On the other hand, the variables introduced by quantified noun phrases cannot be bound as soon as the noun phrases are composed, as in Figure 2, because these variables may need to be used in modifiers composed in subsequent (higher) rule applications. Fortunately, if these non-immediate variable scoping arrangements are expressed structurally, as dominance relationships in the elementary tree structures of some grammar, then a structural restriction on this grammar can be enforced that preserves as many non-immediate scoping arrangements as possible while still preventing an unbounded proliferation of free variables.

The correct scoping arrangements (e.g. for the sentence 'hold a glass under the faucet,' shown Figure 3) can be expressed using ordered sets of parse rules grouped together in such a way as to allow other structural material to intervene. In this case, a group would include a rule for composing a verb and a noun phrase with some associated predicate, and one or more rules for binding each of the predicate's variables in a quantifier somewhere above it (thereby ensuring that these rules always occur together with the quantifier rules dominating the predicate rule), while still allowing rules adjoining prepositional phrase modifiers to apply in between them (so that variables in their associated predicates can be bound by the same quantifiers).

These 'grouped rules' can be formalized using a tree-rewriting system whose elementary trees can subsume several ordered CFG rule applications (or steps in a context-free derivation), as shown in Figure 4. Each such elementary tree contains a rule (node) associated with a logical predicate and rules (nodes) associated with quantifiers binding each of the predicate's variables. These trees are then composed by rewriting operations (dotted lines), which split them up and either insert them between or identify them with (if demarcated with dashed lines) the rules in another elementary tree – in this case, the elementary tree anchored by the word 'under.' These trees are considered elementary in order to exclude the possibility of generating derivations that contain unbound variables or quantifiers over unused variables, which would have no intuitive meaning. The composition operations will be presented in further detail in Section 2.2.

2.2 Semantic composition as tree-rewriting

A general class of rewriting systems can be defined using sets of allowable expansions of some type of object to incorporate zero or more other instances of the same type of object, each of which is similarly expandable. Such a system can generate arbitrarily complex structure by recursively expanding or 'rewriting' each new object, concluding with a set of zero-expansions at the frontier. For example, a context-free grammar may be cast as a rewriting system whose objects are strings, and whose allowable expansions are its grammar productions, each of which expands or rewrites a certain string as a set

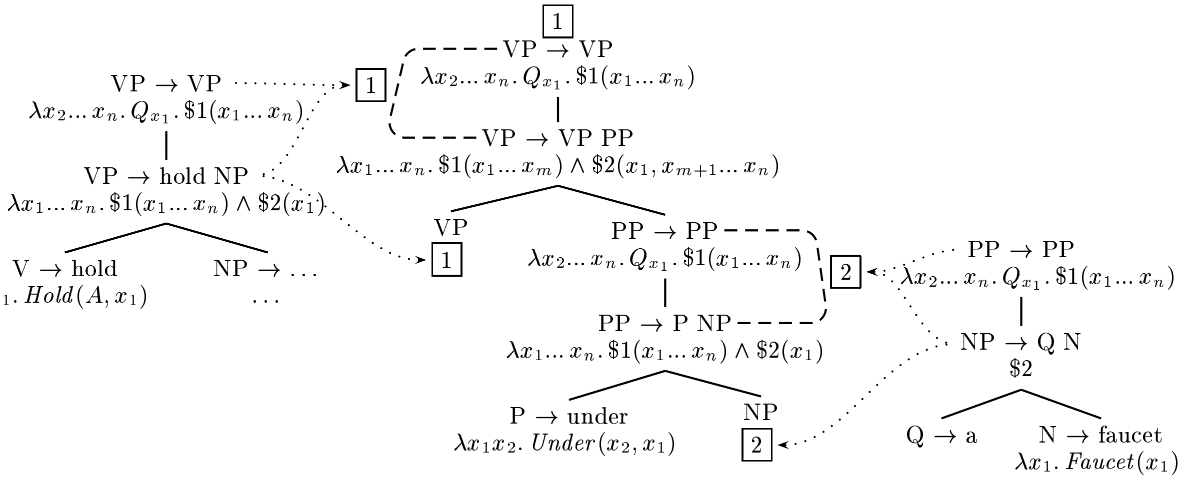

Figure 4: Complete elementary tree for 'under' showing argument insertion sites.

of zero or more sub-strings arranged around certain 'elementary' strings contributing terminal symbols.

A class of *tree*-rewriting systems can similarly be defined as rewriting systems whose objects are trees, and whose allowable expansions are productions (similar to context-free productions), each of which rewrite a tree A as some function f applied to zero or more sub-trees $A_1, \ldots A_s$, $s \geq 0$ arranged around some 'elementary' tree structure defined by f (Pollard, 1984; Weir, 1988):

$$A \rightarrow f(A_1, \ldots A_s) \tag{1}$$

This elementary tree structure can be used to express the dominance relationship between a logical predicate and the quantifiers that bind its variables (which must be preserved in any meaningful derived structure); but in order to allow the same instance of a quantifier to bind variables in more than one predicate, the rewriting productions of such a semantic tree-rewriting system must allow expanded subtrees to identify parts of their structure (specifically, the parts containing quantifiers) with parts of each other's structure, and with that of their host elementary tree.

In particular, a rewriting production in such a system would rewrite a tree A as an elementary tree α_0 with a set of sub-trees $A_1, \ldots A_s$ inserted into it, each of which is first partitioned into a set of contiguous components (in order to isolate particular quantifier nodes and other kinds of sub-structure) using a tree partition function g at some sequence of split points $\langle \vartheta_{i1}, \ldots \vartheta_{ic_i} \rangle$, which are node addresses in A_i (the first of which simply specifies the root).[2] The resulting sequence of partitioned components of each expanded sub-tree are then inserted into α_0 at a corresponding sequence of *insertion site* addresses $\langle \eta_{i1}, \ldots \eta_{ic_i} \rangle$ defined by the rewriting function f:

$$\begin{aligned} f(A_1, \ldots A_s) = \\ \alpha_0[\langle \eta_{11}, \ldots \eta_{1c_1} \rangle, g_{\vartheta_{11}, \ldots \vartheta_{1c_1}}(A_1)] \ldots \\ [\langle \eta_{s1}, \ldots \eta_{sc_s} \rangle, g_{\vartheta_{s1}, \ldots \vartheta_{sc_s}}(A_s)] \end{aligned} \tag{2}$$

Since each address can only host a single inserted component, any components from different sub-tree arguments of f that are assigned to the same insertion site address are constrained to be identical in order for the production to apply. Additionally, some addresses may be 'pre-filled' as part of the elementary structure defined in f, and therefore may also be identified with components of sub-tree arguments of f that are inserted at the same address.

Figure 4 shows the set of insertion sites (designated with boxed indices) for each argument of an elementary tree anchored by 'under.' The sites labeled $\boxed{1}$, associated with the first argument sub-tree (in this case, the tree anchored by 'hold'), indicate that it is composed by partitioning it into three components, each dominating or dominated by the others, the lowest of which is inserted at the terminal node labeled 'VP,' the middle of which is identified with a pre-filled component (delimited by dashed lines), containing the quantifier node labeled 'VP → VP,' and the uppermost of which (empty in the figure) is inserted at the root, while preserving the relative dominance relationships among the nodes in both trees. Similarly, sites labeled $\boxed{2}$, associated with the second argument sub-tree (for the noun phrase comple-

[2]The node addresses encode a path from the root of the tree in which every address $\eta \cdot i$ specifies the i^{th} child of the node at the end of path η.

ment to the preposition), indicate that it is composed by partitioning it into two components – again, one dominating the other – the lowest of which is inserted at the terminal node labeled 'NP,' and the uppermost of which is identified with another pre-filled component containing the quantifier node labeled 'PP $\rightarrow$ PP,' again preserving the relative dominance relationships among the nodes in both trees.

2.3 Shared interpretation

Recall the problem of unbounded variable proliferation described in Section 2.1. The advantage of using a tree-rewriting system to model semantic composition is that such systems allow the application of well-studied restrictions to limit their recursive capacity to generate structural descriptions (in this case, to limit the unbounded overlapping of quantifier-variable dependencies that can produce unlimited numbers of free variables at certain steps in a derivation), without limiting the multi-level structure of their elementary trees, used here for capturing the well-formedness constraint that a predicate be dominated by its variables' quantifiers.

One such restriction, based on the regular form restriction defined for tree adjoining grammars (Rogers, 1994), prohibits a grammar from allowing any cycle of elementary trees, each intervening inside a spine (a path connecting the insertion sites of any argument) of the next. This restriction is defined below:

Definition 2.1 *Let a* spine *in an elementary tree be the path of nodes (or object-level rule applications) connecting all insertion site addresses of the same argument.*

Definition 2.2 *A grammar G is in* regular form *if a directed acyclic graph $\langle V, E\rangle$ can be drawn with vertices $v_H, v_A \in V$ corresponding to partitioned elementary trees of G (partitioned as described above), and directed edges $\langle v_H, v_A\rangle \in E \subseteq V \times V$ from each vertex v_H, corresponding to a partitioned elementary tree that can host an argument, to each vertex v_A, corresponding to a partitioned elementary tree that can function as its argument, whose partition intersects its spine at any place other than the top node in the spine.*

This restriction ensures that there will be no unbounded 'pumping' of intervening tree structure in any derivation, so there will never be an unbounded amount of unrecognized tree structure to keep track of at any step in a bottom-up parse, so the number of possible descriptions of each sub-span of the input will be bounded by some constant. It is called a 'regular form' restriction because it ensures that the set of root-to-leaf paths in any derived structure will form a regular language.

A CKY-style parser can now be built that recognizes each context-free rule in an elementary tree from the bottom up, storing in order the unrecognized rules that lie above it in the elementary tree (as well as any remaining rules from any composed sub-trees) as a kind of promissory note. The fact that any regular-form grammar has a regular path set means that only a finite number of states will be required to keep track of this promised, unrecognized structure in a bottom-up traversal, so the parser will have the usual $\mathcal{O}(n^3)$ complexity (times a constant equal to the finite number of possible unrecognized structures).

Moreover, since the parser can recognize any string derivable by such a grammar, it can create a shared forest representation of every possible analysis of a given input by annotating every possible application of parse rules that could be used in the derivation of each constituent (Billot and Lang, 1989). This polynomial-sized shared forest representation can then be interpreted determine which constituents denote entities and relations in the world model, in order to allow model-theoretic semantic information to guide disambiguation decisions in parsing.

Finally, the regular form restriction also has the important effect of ensuring that the number of unrecognized *quantifier* nodes at any step in a bottom-up analysis – and therefore the number of free variables in any word or phrase constituent of a parse – is also bounded by some constant, which limits the size of any constituent's denotation to a polynomial order of $\mathcal{E}$, the number of entities in the environment. The interpretation of any shared forest derived by this kind of regular-form tree-rewriting system can therefore be calculated in worst-case polynomial time on $\mathcal{E}$.

A denotation-annotated shared forest for the noun phrase 'the girl with the hat behind the counter' is shown in Figure 5, using the noun and preposition trees from Figure 4, with alternative applications of parse rules represented as circles below each derived constituent. This shared structure subsumes two competing analyses: one containing the noun phrase 'the girl with the hat', denoting the entity g_1, and the other containing the noun phrase 'the hat behind the counter', which does not denote anything in the world model. Assuming that noun phrases rarely occur with empty denotations in the training data, the parse containing the phrase 'the girl with the hat' will be preferred, because there is indeed a girl with a hat in the world model.

This formalism has similarities with two ex-

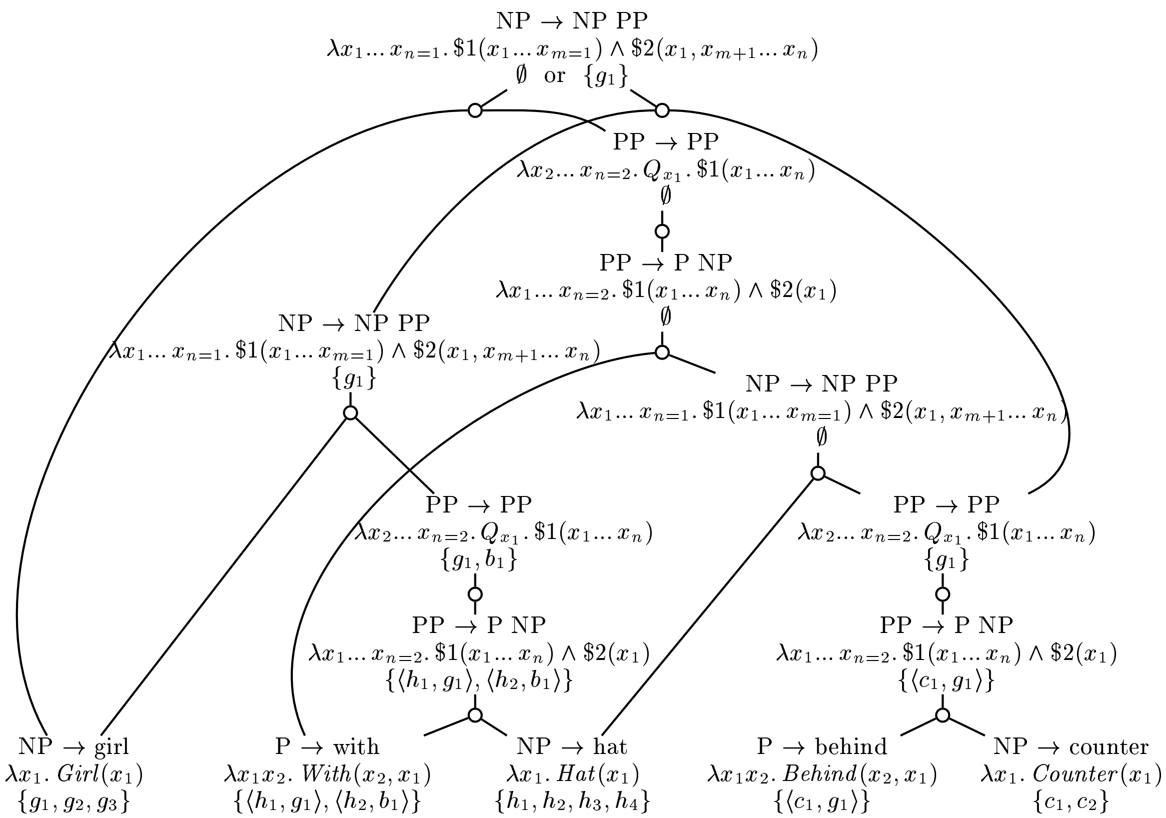

Figure 5: Shared forest for 'the girl with the hat behind the counter.'

tensions of tree-adjoining grammar (Joshi, 1985), namely multi-component tree adjoining grammar (Becker et al., 1991) and description tree substitution grammar (Rambow et al., 1995), and indeed represents something of a combination of the two:

1. Like description tree substitution grammars, but unlike multi-component TAGs, it allows trees to be partitioned into any desired set of contiguous components during composition,

2. Like multi-component TAGs, but unlike description tree substitution grammars, it allows the specification of particular insertion sites within elementary trees, and

3. Unlike both, it allow duplication of structure (which is used for merging quantifiers from different elementary trees).

The use of lambda calculus functions to define decomposable meanings for input sentences draws on traditions of Church (1940) and Montague (1973), but this approach differs from the Montagovian system by introducing explicit limits on computational complexity (in order to allow tractable disambiguation).

This approach to semantics is very similar to that described by Shieber (1994), in which syntactic and semantic expressions are assembled synchronously using paired tree-adjoining grammars with isomorphic derivations, except that in this approach the derived structures are isomorphic as well, hence the reduction of synchronous tree pairs to semantically-annotated syntax trees. This isomorphism restriction on derived trees reduces the number of quantifier scoping configurations that can be assigned to any given input (most of which are unlikely to be used in a practical application), but its relative parsimony allows syntactically ambiguous inputs to be semantically interpreted in a shared forest representation in worst-case polynomial time. The interleaving of semantic evaluation and parsing for the purpose of disambiguation also has much in common with that of Dowding et al. (1994), except that in this case, constituents are not only semantically type-checked, but are also fully interpreted each time they are proposed. There are also commonalities between the underspecified semantic representation of structurally-ambiguous elementary tree constituents in a shared forest and the underspecified semantic representation of (e.g. quantifier) scope ambiguity described by Reyle (1993).[3]

3 Evaluation

The contribution of this model-theoretic semantic information toward disambiguation was evaluated on a set of directions to animated agents collected in a controlled but spatially complex 3-D simulated environment (of children running a lemonade stand). In order to avoid priming them towards particular linguistic constructions, subjects were shown unnarrated animations of computer-simulated agents performing different tasks in this environment (picking fruit, operating a juicer, and exchanging lemonade for money), which were described only as the 'desired behavior' of each agent. The subjects were then asked to direct the agents, using their own words, to perform the desired behaviors as shown.

340 utterances were collected and annotated with brackets and elementary tree node addresses as described in Section 2.2, for use as training data and as gold standard data in testing. Some sample directions are shown in Figure 6. Most elementary trees were extracted, with some simplifications for parsing efficiency, from an existing broad-coverage grammar resource (XTAG Research Group, 1998), but some elementary trees for multi-word expressions had to be created anew. In all, a complete annotation of this corpus required a grammar of 68 elementary trees and a lexicon of 288 lexicalizations (that is, words or sets of words with indivisible semantics, forming the anchors of a given elementary tree). Each lexicalization was then assigned a semantic expression describing the intended geometric relation or class of objects in the simulated 3-D environment.[4]

Walk towards the tree where you see a yellow lemon on the ground.

Pick up the lemon.

Place the lemon in the pool.

Take the dollar bill from the person in front of you.

Walk to the left towards the big black cube.

Figure 6: Sample utterances from collected corpus.

The interface was tested on the first 100 collected utterances, and the parsing model was trained on the remaining utterances. The presence or absence of a denotation of each constituent was added to the label of each constituent in the denotation-sensitive parsing model (for example, statistics were collected for the frequency of 'NP:$-$ $\rightarrow$ NP:+ PP:+' events, meaning a noun phrase that does not denote anything in the environment expands to a noun phrase and a prepositional phrase that do have a denotation in the environment), whereas the baseline system used a parsing model conditioned on only constituent labels (for example, 'NP $\rightarrow$ NP PP' events). The entire word lattice output of the speech recognizer was fed directly into the parser, so as to allow the model-theoretic semantic information to be brought to bear on word recognition ambiguity as well as on structural ambiguity in parsing.

Since any derivation of elementary trees uniquely defines a semantic expression at each node, the task of evaluating this kind of semantic analysis is reduced to the familiar task of evaluating a the accuracy of a labeled bracketing (labeled with elementary tree names and node addresses). Here, the standard measures of labeled precision and recall are used. Note that there may be multiple possible bracketings for each gold standard tree in a given word lattice that differ only in the start and end frames of the component words. Since neither the baseline nor test parsing models are sensitive to the start and end frames of the component words, the gold standard bracketing is simply assumed to use the most likely frame segmentation in the word lattice that yields the correct word sequence.

The results of the experiment are summarized below. The environment-based model shows a statistically significant ($p<.05$) improvement of 3 points in labeled recall, a 12% reduction in error. Most of the improvement can be attributed to the denotation-sensitive parser dispreferring noun phrase constituents with mis-attached modifiers, which do not denote anything in the world model.

Model	LR	LP
baseline model	82%	78%
baseline + denotation bit	85%	81%

4 Conclusion

This paper has described an extension of the semantic grammars used in conventional spoken language

[3]Denotation of competing applications of parse rules can be unioned (though this effectively treats ambiguity as a form of disjunction), or stored separately to some finitie beam (though some globally preferable but locally dispreferred structures would be lost).

[4]Here it was assumed that the intention of the user was to direct the agent to perform the actions shown in the 'desired behavior' animation.

interfaces to allow the probabilities of derived analyses to be conditioned on the results of a model-theoretic interpretation. In particular, a formal restriction was presented on the scope of variables in a semantic grammar which guarantees that the denotations of all possible analyses of an input utterance can be calculated in polynomial time, without undue constraints on the expressivity of the derived semantics. Empirical tests show that this model-theoretic interpretation yields a statistically significant improvement on standard measures of parsing accuracy over a baseline grammar not conditioned on denotations.

References

Tilman Becker, Aravind Joshi, and Owen Rambow. 1991. Long distance scrambling and tree adjoining grammars. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics (EACL'91)*, pages 21–26.

Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL '89)*, pages 143–151.

Alonzo Church. 1940. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(2):56–68.

John Dowding, Robert Moore, François Andery, and Douglas Moran. 1994. Interleaving syntax and semantics in an efficient bottom-up parser. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL'94)*.

Aravind K. Joshi. 1985. How much context sensitivity is necessary for characterizing structural descriptions: Tree adjoining grammars. In L. Karttunen D. Dowty and A. Zwicky, editors, *Natural language parsing: Psychological, computational and theoretical perspectives*, pages 206–250. Cambridge University Press, Cambridge, U.K.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, pages 55–61.

Richard Montague. 1973. The proper treatment of quantification in ordinary English. In J. Hintikka, J.M.E. Moravcsik, and P. Suppes, editors, *Approaches to Natural Langauge*, pages 221–242. D. Riedel, Dordrecht. Reprinted in R. H. Thomason ed., *Formal Philosophy*, Yale University Press, 1994.

Carl Pollard. 1984. *Generalized phrase structure grammars, head grammars and natural langauge.* Ph.D. thesis, Stanford University.

Owen Rambow, David Weir, and K. Vijay-Shanker. 1995. D-tree grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL '95).*

Uwe Reyle. 1993. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics*, 10:123–179.

James Rogers. 1994. Capturing CFLs with tree adjoining grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL '94).*

Stephanie Seneff, Ed Hurley, Raymond Lau, Christine Pao, Philipp Schmid, and Victor Zue. 1998. GALAXY-II: a reference architecture for conversational system development. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP '98)*, Sydney, Australia.

Stuart M. Shieber. 1994. Restricting the weak-generative capability of synchronous tree adjoining grammars. *Computational Intelligence*, 10(4).

David Weir. 1988. *Characterizing mildly context-sensitive grammar formalisms.* Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

XTAG Research Group. 1998. A lexicalized tree adjoining grammar for english. Technical report, IRCS, University of Pennsylvania.