Generation of Paraphrases from Ambiguous Logical Forms

Hadar Shemtov
Stanford University and Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304
shemtov@csli.stanford.edu

Abstract

This paper presents a method for generating multiple paraphrases from ambiguous logical forms. The method is based on a chart structure with edges indexed on semantic information and annotations that relate edges to the semantic facts they express. These annotations consist of logical expressions that identify particular realizations encoded in the chart. The method allows simultaneous generation from multiple interpretations, without hindering the generation process or causing any work to be superfluously duplicated.

1 Introduction

This paper describes a new generation method that produces multiple paraphrases from a semantic input which may contain ambiguities. The method is an extension of the chart based generation algorithm described in Kay (1996). The focus in this presentation is on generating multiple paraphrases and the ability to operate on logical forms that contain more than one semantic analysis. The motivation for this is to enable a situation (particularly in machine translation) where the resolution of ambiguity is postponed to after the generation process. This may open the possibility for considering target language statistics (Knight and Hatzivassiloglou, 1995; Dagan *et al.*, 1991) or more generally for applying other criteria to select the "best" translation, which take into account properties of both languages – for example, prefering ambiguity preserving translations. It may also enable different kinds of interactions between the translation system and the human expert who operates it – for instance, disambiguation by a monolingual in the target language.

The first demonstration of using charts for generation appeared in Shieber (1988). In that paper the emphasis was to show that a uniform architecture can be used for both parsing and generation, however the conception of the chart was limited and the generation algorithm did not appear to be sufficiently attractive. Kay (1996) provides a more general view of the chart structure which is designed to provide for generation advantages comparable to those it provides for parsing. Neumann (1994) proposes another version of a uniform chart architecture where the same data structures are used for both generation and parsing.

In this discussion of chart generation we will focus on one key advantage of the chart structure: the fact that equivalent phrases can fit into larger structures once, regardless of the number of alternatives that they represent. This is achieved by collapsing different derivations that cover the same subset of input (and have the same syntactic potential) under a single edge that represents an equivalence class. This property is the basis for the efficiency gained by using charts as it allows a compact representation in which a polynomial number of edges can potentially encode exponentially many derivations. Thus, the ability to recognize equivalence is an important aspect of chart processing and it is essential that it will be available to the generation process.

We will not describe the underlying generation algorithm in detail but we assume that familiarity with chart parsing is sufficient for understanding the proposed method – the generator can be thought of as a parser that takes logical forms as input and produces strings as analyses. Like a packed parsing forest which represents multiple parsing results, the chart generator produces a "packed generation forest" to represent the various string realizations of the semantics. In the method we propose here, these forests are annotated with information that enables keeping track of the relation between pieces of the semantics and the various phrases that express them. We will concentrate on a detailed description of these annotations as they are a crucial component of our method and they are the major difference between the current proposal and the one described in Kay (1996). Before we do that we will sketch a version of Kay's algorithm, emphasizing data representations rather than algorithmic details. We will also follow Kay in adopting a "flat" representation of event semantics to represent the logical forms (Davidson, 1980; Parsons, 1990). This style of seman-

tics fits the operation of the generation algorithm very well and it is attractive to translation since it allows for flexibility and simplicity with regard to syntactic realization and treatment of structural mismatches between syntax and semantics. The flat structure is also convenient for encoding unresolved ambiguities (Copestake *et al.*, 1996).

2 Kay's Chart Generation Algorithm

In his algorithm, Kay proposes to use two devices to establish which phrases interact and when phrases can be folded together under a disjunctive edge. One device involves indexing edges on semantic variables and another keeps track of which part of the semantics each derivation expresses. In the semantic representation used in the algorithm, each fact is a predicate specifying a relation between events and entities. The events and entities are represented as variables that appear in the predicates and connect the various facts together. For example, the logical form

[chase(e,d,c), dogs(d), young(d), cats(c), young(c)]

denotes a chasing event (e) in which young dogs (d) chase young cats (c). Given this semantics as its input, the generator creates nominal edges with indices d and c as a realization of [dogs(d), young(d)] and [cats(c), young(c)], respectively, and verbal edges with index e as a realization of [chase(e,d,c)]. The packed generation forest encoding the four different realizations of the semantics (obtained by freely choosing between two ways of expressing each of the arguments: "young dogs"/"puppies" and "young cats"/"kittens") is given in figure 1.[1]

Concentrating on the first argument, the constituents which render facts about the "dog" (nodes 1 to 6) are indexed on variable d; nodes 3 and 5 are folded together under node 6 as they are syntactically and semantically equivalent. The semantic equivalence is established on the basis of the indexing variable and the coverage of facts from the logical form.

In parsing, identifying the coverage of the input is straightforward since phrases consist of consecutive items and combine at common end-points; the coverage of each edge is uniquely defined by its string posi-

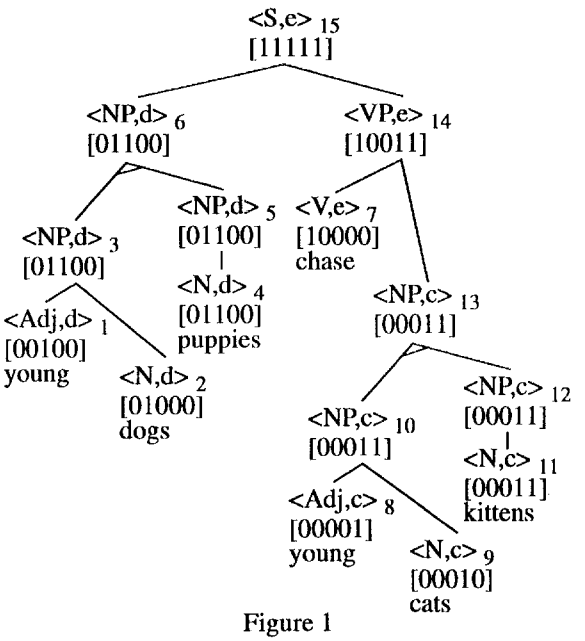

Figure 1

tions. In generation this is not available since the semantics is unordered and the formation of subsets is relatively free – different lexical entries may cover different parts of the input and different syntactic realizations may choose to pack different facts together. Another source of complication comes from the fact that the generation chart encodes multiple paraphrases and we need to guarantee that a piece of semantics will not be expressed more than once.

The mechanism used for keeping track of the semantic coverage of each edge consists of a bit array that represents the set of semantic facts. Each slot in the array corresponds to one fact and indicates whether the fact is expressed by that edge. When edges combine to form a larger constituent, their arrays union together and checked to verify that no fact is duplicated.

The new generation method we propose in this paper is different from Kay's mainly in the criteria for indexing phrases and the mechanism used for determining the semantic coverage. The next sections describe these differences and demonstrate how they can be deployed to enable generation from undisambiguated semantics.

3 Generation with Annotated Charts

We propose a coarser notion of equivalence in order to let more phrases to be folded together. We still use the semantic variables as indices but we do not let the bit arrays be part of the identification of edges. We compensate for this by using a more powerful (and admittedly more complicated) mechanism to relate each constituent to the subset of the semantics it realizes.

[1]Note that the traditional representation of charts (as transition diagrams) is not suitable for generation charts, essentially because of the absence of fixed positions. In order to simplify the exposition, we choose to represent the packed generation forest as an AND-OR tree (in which OR-nodes represent equivalent alternations and AND-nodes represent combination of daughters into larger constituents; OR-nodes are distinguished by the little arcs between their branches). Note that a forest representing multiple paraphrases can be reentrant, as later examples will demonstrate.

The mechanism consists of an array of boolean conditions, each corresponding to one semantics fact. A condition identifies a certain partial path in the packed generation forest; when this path is selected, the corresponding semantic fact is expressed.

In the simple cases, when a constituent expresses the same semantic facts in all of its realizations, the condition can indeed be thought of as binary: if a slot contains 1 the corresponding fact is expressed by the edge and conversely if it is 0. In more complicated cases, when an edge has different realizations that cover different parts of the semantics, the indications in the arrays are given as boolean expressions composed of propositional variables. Each disjunctive edge in the chart is annotated with a set of such variables, each of which mutually exclusively defines a particular alternative derivation of that edge. These propositional variables compose into larger boolean expressions that define derivations of larger structures. For a general explanation of the method for using boolean expressions to handle disjunctions, see Maxwell and Kaplan (1989).

The next example shows a chart with semantic arrays and exemplifies how the conditions appearing in their slots control realizations of the input. Consider the following logical form:

[dog(d), plural(d), big(d), bark(e,d), loud(e)]

and the chart (forest) that would be constructed from it by the generation algorithm:

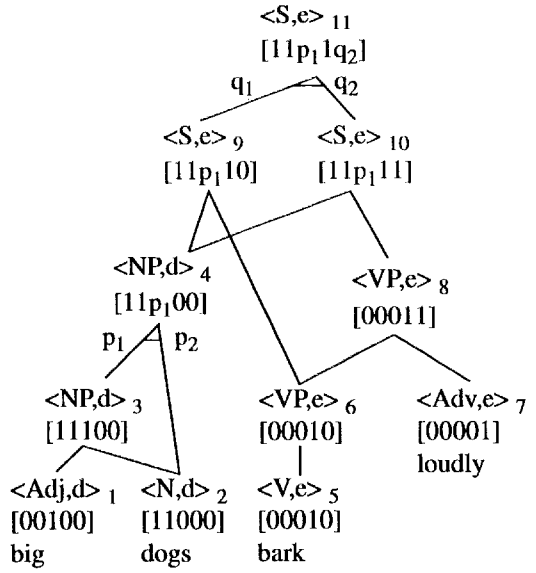

Figure 2

In this drawing, the branches of the OR-nodes are labeled with propositional variables and below each edge is the array that indicates its coverage. For instance, node 2, which expresses "dogs," covers the first 2 facts, hence its array is [11000]; node 5 which expresses "bark" contributes the 4th fact [bark(e,d)] and accordingly, its array is [00010]. Nodes 4 and 11 are disjunctive with choices represented by the proposition variables $p_{1,2}$ and $q_{1,2}$ respectively. The NP of edge 4 can be realized as "dogs" if p_2 is chosen or as "big dogs," if p_1 is chosen. This is reflected in the third slot of the array. It indicates that the third semantic fact is expressible in condition p_1. Likewise, the top-most S (node 11) is disjunctive since there are two ways to form a sentence: either using the VP of node 6 or the one of node 8, which also expresses the fifth fact about the barking event being loud. This explains the reason that expression of the fact [loud(e)] is conditioned on the choice q_2 (the 5th slot of the array in node 11). The two conditions taken together mean that a complete expression of the semantic input is conditioned on both p_1 and q_2 being the choices in the relevant disjunctions.

This example begins to show how the various components of the representation control the generation process. Before we continue with examples of more complex boolean conditions, we explain how the boolean arrays are constructed and what exactly is their logical interpretation.

3.1 Construction of the Boolean Arrays

In addition to the syntactic composition, the boolean arrays of the daughter constituents union to form the semantic array of the resulting mother constituent. Usually one daughter will have an array like [...α_i0...], the other [...0β_j...] and their combination will yield [...$\alpha_i\beta_j$...]. However, if both daughters express a particular semantic item the boolean expressions of the corresponding slots need to be disjoined (from the point of view of the mother they are alternative renditions). However, to avoid expressing the same facts more than once, a further constraint is required to guarantee that only one of the disjuncts eventually get chosen. This constraint is the negation of the conjunction of the two conditions. So, if we combine [...α_i0...] with [...$\beta_j\gamma_j$...] the result is [...($\alpha_i|\beta_j$)γ_j...] and the negative condition ~(α_i&β_j) is added as a filter to the mother node. Whenever this node is traversed, the constraint needs to be honored. Note that the negative constraints are not composed into the boolean expressions. For a more elaborate explanation of this device, see Shemtov (1996).

The meaning of combined conditions is the following. A disjunction indicates that there are multiple but mutually exclusive ways of expressing a certain semantic fact. In the array [...($\alpha_i|\beta_j$)...] the fact corresponding to the given slot can be rendered either by choosing the *i*th branch of the α disjunction or the *j*th branch of the β disjunction. A conjunction defines a

part of a certain path in the forest. It means that at two (or more) different nodes, only certain combinations of branches can be selected. In the array [...(α_i&β_j)...] we get a situation where in one OR-node (the α disjunction) we need to select the *i*th branch and in another (the β disjunction) we need to choose the *j*th branch.

Another issue that is solved through the logical interpretation of the conditions is determining that the whole input is consumed. In parsing this is straightforward: there has to be a top-node from string position *0* to string position *n*. In the generation scheme developed here, this is much more complicated. Facts can be expressed only under certain conditions and it needs to be verified that the conditions are honored in a mutually consistent way. To determine whether all the semantic facts are expressed, the boolean conditions from all the slots in the array of the top node are conjoined and the result is checked for satisfiability. If the result is not satisfiable (no consistent assignment of truth values) or if it is not consistent with the negative constraints, then there is no path in the derivation graph that corresponds to an expression of all the facts. Admittedly, computing a satisfiable assignment to the various propositional variables can be hard (exponential complexity in the general case), however certain computational properties which are likely to exist (independence between sets of variables) will tend to make the computation much more efficient.

3.2 Paraphrases

Just as a parsing chart excels in compact representation of multiple interpretation of a single string, the generation chart is designed to represent multiple (string) realizations of the semantic interpretation and compute them at a minimal cost. As the following example demonstrates, the explicit encoding of conditions in which each fact is expressed provides a powerful way of controlling the realizations of the various paraphrases. It also provides a way for verifying that they do not overlap and express certain facts superfluously. Let us assume that the verbs "enter" and "rush" both decompose as movement verbs. The former would be represented as [move(e,agent), into(e,loc)] and the latter as [move(e,agent), quick(e)]. Also let us assume that the meaning of a PP headed by "into" is [into(e,loc)] and that [quick(e)] is also the semantics of the adverb "quickly." With that, consider the following logical form

[John(j), move(e,j), into(e,r), room(r), quick(e)]

and the packed generation forest representing its various derivations (figure 3). The interesting action is in the fifth slot. [quick(e)] can be expressed by satisfying the condition q_1&p_1&r_2 which means choosing the left

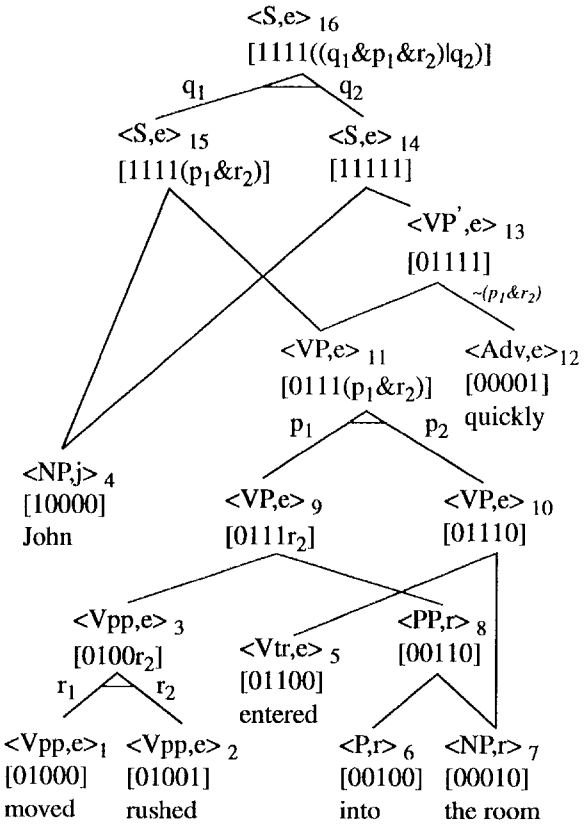

Figure 3

branches in nodes 16 and 11 and the right branch at node 3. This path corresponds to the sentence "John rushed into the room." Another expression can be obtained by choosing q_2 at node 16; this leads to node 13 on whose right branch the adverb "quickly" expresses [quick(e)]. Now, this situation is interesting because this fact is already contained in one of the branches of node 11, as we have already seen. To avoid expressing it twice, a further negative constraint is placed on node 13 which requires p_1&r_2 to be false. The constraint excludes the path that leads to a selection of the verb "rush" but it allows a choice of p_2, which means that "enter" can be used to yield "John entered the room quickly." It also allows a choice of the verb "move" since p_1&r_1 represents a valid path. This way the sentence "John moved into the room quickly" is realized.

This example demonstrates how multiple paraphrases are constructed out of a variety of lexical entries and syntactic constructions and how a record is kept relating the different phrases to the subsets of the semantic facts that they express. It shows that the generation method is sensitive to the particular lexicalization patterns that languages use to encode divergent parts of the semantics.

4 Generation from Ambiguous Semantics

The logical encoding of the boolean conditions may seem complex and indeed simpler solutions have been proposed to encode the semantic coverage (in Kay's algorithm for instance). However, the aim of the generation method we advocate here goes beyond rendition of fully specified semantics.

One translation situation that the annotated chart approach can address very simply has to do with optional and defeasible specifications. In many situations there may be certain specifications in the input (discourse consideration, indication of preferences, etc.) that may not be crucial to the adequacy of the resulting expressions. For instance, in translation one might prefer to maintain the source language subject as the target language subject but be willing to accept a translation which violates this if generation would otherwise fail. This can happen when the source expression is passive but the corresponding target language verb does not passivize. Similarly, certain psychological verbs come in pairs ("fear/frighten", "like/please" etc.) but not in all languages, therefore a specification to express a particular argument as the discourse topic might lead to a failure. For example, translating "John likes it" into Spanish most naturally comes out as "it pleases John." The idea is that in such cases the generator will attempt to find an expression that conveys (or honors) all the specifications, but if such an expression is not admitted by the grammar it would still produce a grammatical result covering the crucial parts of the input.

A more interesting problem that a chart with boolean conditions can address is how to use ambiguous semantics as an input to the generation process. Given that exhaustive disambiguation is not always possible, the idea is that the choice among the source language analyses will be delayed and the whole set of semantic interpretations will comprise the input to the generation process. The motivation is to gain more information from the target language in order to improve the quality of the choice. The crucial advantage that the proposed generation method provides is that it enables considering all of the semantic interpretations "at once," avoiding the massive duplicated effort that would result from enumerating the logical forms and considering each one of them individually.

The next two simplified examples demonstrate how logical forms which contain disjunctions can be processed by the generator and how the rich logical annotations relate the various paraphrases to the alternations in the semantics. The first example demonstrates a disjunction resulting from a structural ambiguity. The expression "hydraulic oil filter" lends itself to two different bracketings, corresponding to "filter for hydraulic oil" and "hydraulic filter for oil." These two interpretations are given in the following disjunctive logical form:

[filter(f), oil(o), {hydraulic(o) | hydraulic(f)}]

Figure 4 shows the packed generation forest that encodes the two (incidentally identical) strings that express this piece of semantics.

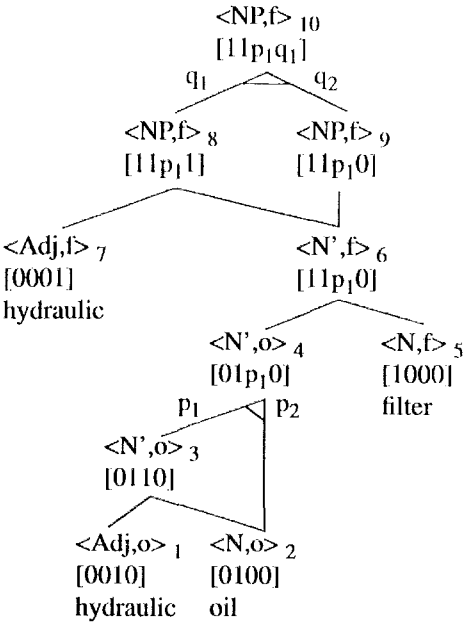

Figure 4

The generation from a disjunctive input proceeds just as before, as if the disjunction is ignored and all the semantic facts are given equal status. Then, when the results of the generation are to be enumerated, the logical structure of the input reappears and affects the interpretation of the boolean array. In this example, we know that either the third fact or the fourth fact (but not both) can be expressed. Accordingly, we allow either the third or the fourth boolean condition to be satisfied. If we choose to satisfy the former, we let p_1 be true and q_1 be false. This forces a traversal of nodes 9 and 3 which amounts to generating from [filter(f), oil(o), hydraulic(o)]. If on the other we choose to express the other interpretation, we reverse the conditions. This requires a selection of the left branch in node 10 (q_1) which means that [hydraulic(f)] gets expressed in node 7. At node 4 we refrain from expressing [hydraulic(o)] since we set p_1 (the condition in the third slot) to false. This way we reconstruct the logical structure of the disjunctive logical form and select one interpretation at a time from the set of possible paraphrases.

The next example shows how an NP is generated from a specification that results from a lexical ambiguity. Let us consider the following logical form that could be produced from analyzing "little dog" in a lan-

guage that interprets "little" as an ambiguous adjective denoting either smallness in size or youngness in age.

[dog(d), {small(d) | young(d)}

We assume that this semantics licenses "small dog," "young dog" and "puppy" (but not "young puppy" or "small puppy"). Figure 5 shows the generation forest that encodes these renditions of the input.

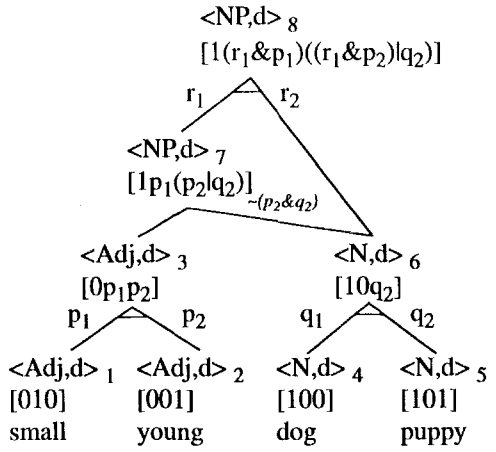

Figure 5

Node 3 merges two different adjectives which are indexed on the same variable but express two different facts; Node 6 merges two nominal phrases with compatible but not completely overlapping meanings. Now, if our goal is to enumerate the paraphrases corresponding to the first interpretation, we satisfy the condition in the second slot [small(d)] and dissatisfy the condition in the third slot [young(d)]. As a result, we select the left branches of nodes 8 and 3 so as to satisfy $r_1\&p_1$. Note that at node 6 we can only choose the left branch because otherwise the condition of the third slot would also be satisfied, contrary to the mutually exclusive nature of the semantic alternation. When the goal is to generate the second interpretation, we reverse the conditions and try to satisfy $(r_1\&p_2)|q_2$. If r1 and p2 are set to true we get "young dog." If q_2 is selected we choose the right branches of nodes 8 and 6 and get "puppy."

These two examples demonstrate how we manipulate the boolean conditions of the semantic coverage arrays to allow generation from a disjunctive input and still gain the benefits of the chart generation algorithm.

5 Future Work

The method we propose in this paper can be deployed as an infrastructure for solving certain other problems in generation and translation. In future work (Shemtov, 1996) we intend to use the ideas developed here to tackle the problem of ambiguity preserving translation. Our approach is to take a parsing chart as an input, read from it an ambiguous logical form encoding multiple source language interpretations and then use it to create a generation chart encoding multiple target language strings. A separate process will then search for strings that express more than one interpretation; If such strings are found, we say that the ambiguity of the source language is preserved by the target language. We hope that by using this approach it will be possible to avoid certain types of disambiguations altogether.

Acknowledgments

I wish to thank Martin Kay, John Maxwell and Ronald Kaplan for their interest, comments and encouragement. I am also indebted to the anonymous reviewers of this paper.

References

Copestake, Ann; Flickinger, Dan; Malouf, Robert; Riehemann, Susanne and Sag, Ivan (1996). "Translation Using Minimal Recursion Semantics." In *Proceedings, 6th International Conference on Theoretical and Methodological Issues in Machine Translation.*

Dagan, Ido; Itai, Alon and Schwall, Ulrike (1991). "Two languages are more informative than one." In *Proceedings, 29th annual meeting of the ACL.* 130-137.

Davidson, David (1980) *Essays on Actions and Events.* Oxford: The Clarendon Press.

Kay, Martin (1996). "Chart Generation." In *Proceedings, 34th annual meeting of the ACL.*

Knight, Kevin and Hatzivassiloglou, Vasileios (1995). "Two-Level, Many-Paths Generation." In *Proceedings, 33rd annual meeting of the ACL.* 252-260

Maxwell, John T. III and Kaplan, Ronald M. (1989). "A Method for Disjunctive Constraint Satisfaction". In *Current Issues in Parsing Technology,* edited by Masaru Tomita, 173-190. Kluwer.

Neumann, Günter (1994). *A Uniform Computational Model for Natural Language Parsing and Generation.* Ph.D. Thesis, University of the Saarland.

Parsons, Terence (1990). *Events in the Semantics of English.* Cambridge, Mass.: MIT Press.

Shemtov, Hadar (1996). *Ambiguity Preserving Translation.* Ph.D. Thesis (in preparation), Stanford University.

Shieber, Stuart (1988). "A Uniform Architecture for Parsing and Generation." In *Proceedings, COLING-88.* 614-619.