M. P. Harper, L. H. Jamieson, C. B. Zoltowski, and R. A. Helzerman, "Semantics and constraint parsing of word graphs," Proc. ICASSP'93, pp. 63 - 66, April 1993.

Semantics and Constraint Parsing of Word Graphs

Mary P. Harper, Leah H. Jamieson, Carla B. Zoltowski, and Randall A. Helzerman
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

Abstract

We have developed a constraint-based parser capable of processing a word graph containing multiple sentence hypotheses. When syntactic constraints are applied to a word graph, our parser is able to prune the graph of many ungrammatical sentence hypotheses and limit the possible parses of the remaining sentences. However, in many cases syntactic information alone is insufficient for selecting a single sentence hypothesis from a word graph. Hence, we have added semantic constraints to our parser to further limit ambiguity. In this paper, we review the constraint parsing algorithm and then provide a simple example illustrating how syntactic and semantic features can be used to prune word candidates from a word graph and eliminate incorrect parses for the remaining sentences. We also report on the effectiveness of syntactic and semantic constraints for reducing the ambiguity of word networks.

1 INTRODUCTION

We [4] extended the Constraint Dependency Grammar (CDG) parser developed by Maruyama [1, 2] to process a graph of sentence hypotheses, as would result from spoken input, rather than single sentences, as would result from textual input. We also developed a grammar containing 3 roles, 11 categories, 70 labels, 100 unary constraints, and 200 binary constraints, capable of parsing a wide variety of sentences including statements, yes-no questions, commands, and wh-questions. This grammar was constructed to demonstrate that syntactic and lexical constraints can be used quite effectively to eliminate ungrammatical sentences from a word graph. The constraint-based framework is appealing for a number of reasons:

1. It supports a variety of knowledge sources (e.g., syntactic, semantic, prosodic) in a uniform framework.
2. It has the ability to handle the syntactic irregularities common in spontaneous speech.
3. It is able to support the use of context when determining the meaning of a sentence (especially to reduce ambiguity).
4. It is able to operate directly on word graphs produced by a speech recognition front end.
5. It is amenable to parallel implementation [3].

In this paper, we modify our parsing framework to incorporate semantic constraints into the set of knowledge sources used to parse word networks. First, we review how the parsing algorithm works and then describe a simple example illustrating how syntactic and semantic features can be used to prune word candidates from a word graph and eliminate incorrect parses for the remaining sentences. We also report on the effectiveness of syntactic and semantic constraints for reducing the ambiguity of word networks constructed for N-best sentence hypotheses provided by BBN [5] from the ATIS database (Air Travel Information System).

2 SLCN PARSING

We have adapted the CDG constraint network to handle multiple word candidates, calling it a Spoken Language Constraint Network (SLCN). This approach allows us to efficiently process multiple sentence hypotheses while maintaining the flexibility of the CDG parsing method. Figure 1 depicts an SLCN derived from a word graph constructed for the sentence hypotheses: *A fish eat*[sic] and *Offices eats*[sic].

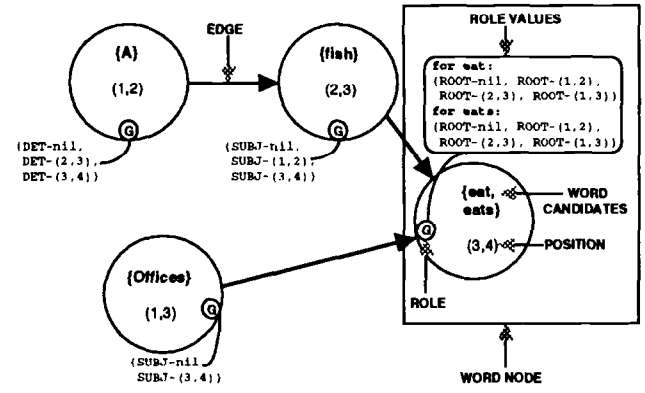

Figure 1. An SLCN constructed from a word graph.

By representing these hypotheses in a word graph, we are also able to process additional sentences (i.e., *A fish eats* and *Offices eat*) not present in the list of hypotheses, one of which might be the correct utterance. *Word nodes* contain information on the *position* of a word's utterance (represented as an integer tuple (b, e), where b and e represent the time at which the word begins and ends, respectively), a set of *roles* (which indicate the various functions the word fills in a sentence), and a list of all *word candidates* with the same *position*. Also, *edges* join word nodes that can be adjacent in a sentence hypothesis. Though two roles are required to write a grammar that is at least as expressive as a CFG [1], our example depicts a single role to simplify the discussion. The depicted role is the *governor* role,

0-7803-0946-4/93 $3.00

which represents the function that a word fills given that it modifies its head. For example, given that the head of a noun phrase is a noun, the word *the* has the function of a determiner (i.e., DET) when it modifies the head noun.

To parse an SLCN, each role for every word is initially assigned all *role values* allowed by each of the word candidates for each of its parts of speech. A role value consists of a label (the function the word can serve, e.g., DET) and a modifiee (the position of another word which can be in at least one of the same sentences as the word in question, or nil). Because more than one word candidate can appear in a word node and each candidate can have more than one part of speech, each role value must keep track of its word and part of speech. There are $p*q*n = O(n)$ possible role values (where p, the number of roles per word, and q, the number of different labels, are grammatical constants and n is the number of words) for each of the n word candidates in the word graph, giving $O(n^2)$ role values altogether and requiring $O(n^2)$ time to generate. Figure 1 shows the initialization of the SLCN, if *a* is a determiner with the label DET, *fish* and *offices* are nouns with the label SUBJ, and *eat* and *eats* are verbs with the label ROOT.

Once the role values are enumerated, constraints are applied to eliminate the ungrammatical role values. A constraint is an *if-then* rule which must be satisfied by the role values. First, *unary constraints* (i.e., constraints with a single variable) are applied, requiring $O(k_u * n^2)$ time (where k_u is the number of unary constraints). To apply a unary constraint, each role value for every role is examined to ensure that it obeys the constraint. A role value violates a constraint iff it causes the antecedent of the constraint to evaluate to TRUE and the consequent to evaluate to FALSE. A role value which violates a unary constraint is eliminated from its role. For example, if the following unary constraint is applied to figure 1, then the role value DET-nil would be eliminated from the governor role for *a*. The less-than predicate, $(<\ (b1,\ e1)\ (b2,\ e2))$, returns true if $e1 < b2$.

```
;; A DET must modify something to its right.
;; NIL as a modifiee is eliminated.
(if (equal (label x) DET)
    (< (position x) (modifiee x)))
```

Figure 2 depicts the SLCN after two additional unary constraints are applied requiring ROOTs to have a nil modifiee and SUBJs to modify words after them.

Figure 2 also illustrates how the CN is prepared for the propagation of *binary constraints*. Binary constraints contain two variables and determine which pairs of role values can legally coexist. *Arcs* are added to the SLCN to keep track of which pairs of role values can legally coexist given the binary constraints. All roles within the same word node are joined with an arc; however, roles in different word nodes are joined with an arc iff they can be members of at least one common sentence hypothesis (i.e., they are connected by a path of directed edges). To construct the arcs and arc matrices for an SLCN, it suffices to traverse the graph from left to right and string arcs from each of the current word node's roles to each of the preceding word node's roles (where one node precedes another iff there is a directed edge from the first to the second node) and to each of the roles that the preceding node's roles have arcs to. For example, there should be an arc between the roles for *a* and *fish* in figure 2 because they are located on a path from the beginning to the end of the sentence *the fish* {*eats, eat*}. However, there should not be an arc between the roles for *a* and *offices* since they are not found in any of the same sentence hypotheses.

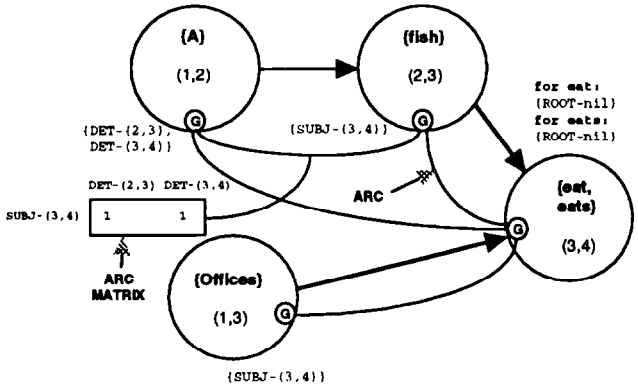

Figure 2. An SLCN prior to binary constraint propagation.

Each of the arcs has an associated *arc matrix*, whose row and column indices are the role values from its two roles. The elements of the arc matrices can hold either a **1** (indicating that the two role values which index it can legally coexist) or a **0** (indicating that the role values cannot legally coexist). Initially, all entries in the matrices are set to **1** except for those entries indexed by role values pointing to words not contained in the sentence hypothesis supporting the arc. Only one of the arc matrices is depicted in figure 2. After the arc matrices are constructed in $O(n^4)$ time, the binary constraints are applied to the pairs of role values that represent the indices for matrix entries, requiring $O(k_b * n^4)$ time (where k_b is the number of binary constraints). If a binary constraint fails for a pair of role values then they cannot coexist in the same sentence, which is indicated by setting the entry in the matrix to zero. Figure 3 shows the state of the SLCN from figure 2 after the following constraint is propagated.

```
;; A DET must modify a SUBJ.
(if (and (equal (label x) DET)
         (equal (label y) SUBJ))
    (equal (modifiee x) (position y)))
```

Following the propagation of binary constraints, the network could still contain role values that would never be legal role values in a parse for the sentence. To determine whether a role value is still supported by an arc matrix, each of the matrices on the arcs incident to the role must be checked to ensure that the row (or column) indexed by the role value contains at least a single **1**. If any arc matrix contains a row (or column) of **0**s for the role value, then that role value cannot coexist with any of the role values for the other role and so the role value is not supported by the arc. In a constraint network for single sentences, *filtering* eliminates each of the unsupported role values for an arc from its role and all of the arcs associated with the role. However, filtering in an SLCN is complicated by the fact that the elimination of a role value on one arc associated with a role may not cause the role value to be eliminated for

the other arcs associated with that role. For example, even if the role value for *eat* in figure 3 is disallowed by the singular subject *fish*, those role values cannot be eliminated since they are supported by *offices*, the subject in a different hypothesis.

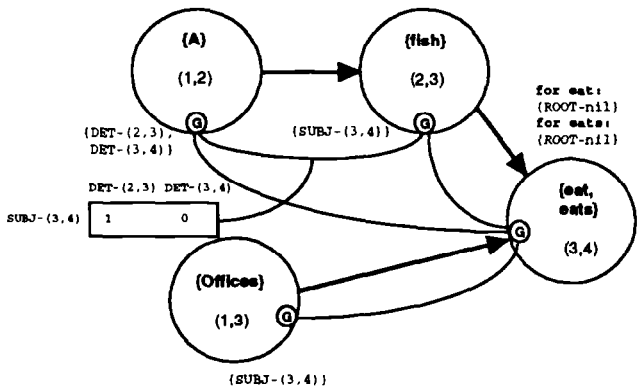

Figure 3. An SLCN after binary constraint propagation.

If **arc(Role1,Role2)** does not support a role value **r** in **Role1** then the role value should only be eliminated from arcs of the form **arc(Role1,X)** when **arc(Role1,Role2)** and **arc(Role1,X)** are members precisely the same set of sentence hypotheses. For example, given that DET-(3,4) is not supported by the arc matrix for the arc between the governor roles of *a* and *fish*, it should also be eliminated by the matrix associated with the arc between the governor roles for *a* and {*eat, eats*} since it is a member of the same sentence hypothesis as the non-supporting arc. Since none of the arcs supports the role value DET-(3,4), it should be eliminated from the governor role of *a*. If all of the role values in a role for a particular word candidate are eliminated by filtering, then that word candidate is no longer a supported word, and if all of the word candidates for a word node are unsupported, then the word node is also unsupported. Furthermore, word nodes which are no longer members of a legal sentence hypothesis because the only word they are adjacent to is unsupported will, through filtering, lose support. Filtering of an SLCN requires $O(n^4)$ time [4, 3]. Following filtering, the remaining role values form parse graphs for the remaining sentences.

3 SLCN PARSING AND SEMANTIC FEATURES

Next, we discuss how syntactic and semantic features can be used to eliminate ungrammatical and semantically anomalous sentence hypotheses from the SLCN in figure 4. To correctly utilize features in constraints, each role value must be assigned a single feature value, not a set of values. If there is more than one feature value possible for a word, then the role values are duplicated for each of them (just as we create role values for each part of speech for each word, we also create role values for each feature for a word). If there are two feature types (say number/person and case) to be used in constraints for a grammar, then the role values will have to be duplicated and assigned feature values from the cross product of the features' values. This could possibly lead to a combinatorial explosion of role values. Fortunately, there is an excellent strategy for keeping the number of role values down. The basic idea is to store the sets of feature values with a single role value and then to initially propagate only constraints without feature tests, eliminating many of the ungrammatical role values. When constraints that test a particular feature are going to be applied to the SLCN, a role value with multiple feature values is duplicated and assigned a single feature value. The corresponding feature constraints should eliminate many of the duplicated role values. The process of selecting a feature and duplicating role values continues until all feature constraints have been propagated.

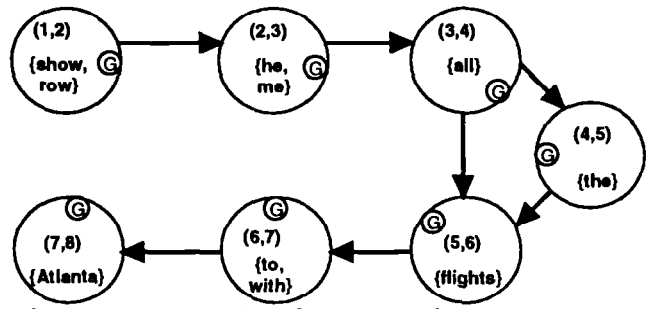

Figure 4. SLCN for the semantics example.

This analysis strategy requires several stages of constraint propagation. Consider how feature analysis can be used to reduce the ambiguity of the SLCN in figure 4. First, all syntactic unary and binary constraints without feature tests are propagated, eliminating many of the role values before feature information is used. Below is a table showing the remaining role values:

Position:	Word:	Role Values:
(1,2)	show	C_ROOT-nil
		;; A Command verb
	row	C_ROOT-nil
		;; A Command verb
(2,3)	me	IOBJ-(1,2)
		;; Indirect Object for node (1,2)
	he	IOBJ-(1,2)
		;; Indirect Object
(3,4)	all	DET_MOD-(4,5)
		;; A Special DET
(4,5)	the	DET-(5,6)
		;; Determiner for node (5,6)
(5,6)	flights	DO_IOBJ-(2,3)
		;; A DO in an S with an IOBJ
(6,7)	to	N_PP-(5,6)
		;; PP modifying a Noun
		V_PP-(1,2)
		;; PP modifying a Verb
	with	N_PP-(5,6)
		;; PP modifying a Noun
		V_PP-(1,2)
		;; PP modifying a Verb
(7,9)	Atlanta	PP_OBJ-(6,7)
		;; Object of a Preposition

Next, case features can be used to reduce the ambiguity in the network. Since *me* is objective case, *he* is subjective case, and both *Atlanta* and *flights* are common case (i.e., they can go anywhere), each of these role values can be assigned a single case feature. If a constraint is propagated which requires a role value with the label IOBJ to have either objective or common case, then the single role value for the word *he* would be eliminated because of case incompatibility.

Subcategorization features can also be used to reduce the ambiguity in the network. Suppose that *show* and *row* can subcategorize for the objects indicated below:

Position:	Word:	Role Values:	Subcategorization:
(1,2)	show	C_ROOT-nil	Indirect & Direct Object
	show	C_ROOT-nil	Direct Object
	row	C_ROOT-nil	Direct Object

We have developed constraints for an SLCN which use three roles to enforce verb subcategorization requirements. These constraints eliminate the single role value for *row* and the direct object role value of *show*, since neither accounts for the second object.

Many impossible parses for the sentences in the word graph have been eliminated by syntactic constraints and two word candidates have been eliminated by syntactic feature tests. Semantic constraints can also be used to eliminate both word candidates and parses. After propagating the syntactic feature constraints, node (6,7) has two word candidates, each with two possible parses (the prepositional phrases can either attach to *flights* or to *show*). To further disambiguate this sentence, we assign semantic features to several key words.

Position:	Word:	Role Values:	Sem. Features:
(1,2)	show	C_ROOT-nil	demo_event
(5,6)	flights	HEAD_NOUN-(3,4)	move_event
(6,7)	to	N_PP-(5,6)	to_loc
		N_PP-(5,6)	to_time
		N_PP-(5,6)	to_poss
		V_PP-(1,2)	to_loc
		V_PP-(1,2)	to_time
		V_PP-(1,2)	to_poss
	with	N_PP-(5,6)	coagent
		N_PP-(5,6)	sub_event
		N_PP-(5,6)	to_poss
		V_PP-(1,2)	coagent
		V_PP-(1,2)	sub_event
		V_PP-(1,2)	to_poss
(7,9)	Atlanta	PP_OBJ-(6,7)	location

Semantic feature constraints can now be used to completely disambiguate the word graph. The first constraint uses the semantic type of the head word in the object of the preposition to restrict the semantic type of the preposition.

```
;; A location that is an object of a preposition
;; modifies a locative preposition.
(if (and (equal (lab x) PP_OBJ)
         (equal (modifiee x) (position y))
         (equal (semantic_type x) location))
    (member (semantic_type y) {to_loc from_loc at_loc}))
```

After this constraint is propagated, the only remaining role values for the word *to* are the two with to_loc as their semantic feature. Because the semantic type of each of the role values for *with* are incompatible with the semantic type of its object, it is eliminated as a word candidate. A final semantic constraint can now restrict the type of word to which a to_loc preposition can attach:

```
;; A to_loc preposition modifies a move_event.
(if (and (member (lab x) {N_PP V_PP})
         (equal (mod x) (pos y))
         (equal (semantic_type x) to_loc))
    (equal (semantic_type y) move_event))
```

This constraint eliminates the role value V_PP-(1,2), giving a completely disambiguated word network.

We have conducted a simple experiment to compare the effectiveness of syntactic and semantic constraints for reducing the ambiguity of word networks constructed from sets of BBN's N-best sentence hypotheses [5] from the ATIS database (Air Travel Information System). For this experiment, we selected twenty sets of 10 N-best sentence hypotheses for three different types of utterances: a command, a yes-no question, and a wh-question. The lists of the N-best sentences were converted to word graphs in which the duration of each node was determined by maintaining a syllable count through the utterance. We then determined for each eliminated word candidate whether a syntactic constraint or a semantic constraint was responsible for the deletion. On the average, syntactic constraints eliminated 3.11 word candidates per SLCN. In contrast, semantic constraints were directly responsible for eliminating .66 word candidates per SLCN. Furthermore, the only constraints responsible for eliminating word nodes were syntactic. However, semantic constraints are very effective at reducing the number of remaining parses following syntactic constraint propagation.

Current spoken language recognition systems are not as accurate as humans, in part, because they do not utilize the wide range of information that people do when understanding speech. SLCN parsing enables the incorporation of syntactic and semantic knowledge into the speech recognition process. We hope that further investigations along these lines will result in improved recognition accuracy.

4 ACKNOWLEDGEMENTS

This work was supported in part by Purdue Research Foundation, NSF grant number IRI-9011179, and NSF Parallel Infrastructure Grant CDA-9015696. We thank BBN for providing us with the N-best lists of sentences.

References

[1] H. Maruyama. Constraint dependency grammar and its weak generative capacity. *Computer Software*, 1990.

[2] H. Maruyama. Structural disambiguation with constraint propagation. In *The Proceedings of the Annual Meeting of ACL*, 1990.

[3] R. Helzerman, M.P. Harper, and C.B. Zoltowski. Parallel parsing of spoken language. In *Proceedings of the Fourth Symposium on the Frontiers of Massively Parallel Computation*, October 1992.

[4] C.B. Zoltowski, M.P. Harper, L.H. Jamieson, and R. Helzerman. Parsec: A constraint-based framework for spoken langauge understanding. In *Proceedings of the International Conference on Spoken Langauge Understanding*, October 1992.

[5] R. Schwartz and Y-L. Chow. The N-best algorithm: An efficient and exact procedure for finding the N most likely sentence hypotheses. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, April 1990.

[6] J. Allen. *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, Menlo Park, CA, 1987.