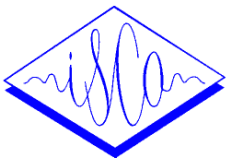

2nd International Conference on
Spoken Language Processing (ICSLP 92)
Banff, Alberta, Canada
October 12-16, 1992

PARSEC: A Constraint-based Framework for Spoken Language Understanding

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

Abstract

We have extended Maruyama's [5, 6, 7] constraint dependency grammar (CDG) to process a lattice or graph of sentence hypotheses instead of separate text strings. A post-processor to a speech recognizer producing N-best hypotheses generates the word graph representation, which is then augmented with information required for parsing. We will summarize the CDG parsing algorithm and then describe how the algorithm is extended to process a word graph on a single processor machine.

1 Introduction

The most successful of the current speech recognition systems which process continuous speech for a limited (1000 word) vocabulary are those which utilize hidden Markov models (HMM). Most systems utilizing this approach (e.g., [4, 10])) have reduced recognition errors by incorporating some language information (syntactic and semantic) directly into the HMM to reduce perplexity, but since the goal of these systems is recognition, not understanding, no structural analysis of the utterance is constructed. Instead, the output of such systems is an ordered list of the N most likely sentence hypotheses (where N is a constant usually less than 100) [9, 11]. If understanding becomes the goal, such systems must pass the sentence hypotheses through a natural language parser as a first step toward producing meaning representations. A context-free grammar (CFG) parser would require $O(n^3)$ time to process each sentence hypothesis containing n words.

Processing each sentence hypothesis individually is inefficient since the sentence hypotheses often differ only slightly from each other. Furthermore, a list of sentence hypotheses is not the most compact representation to provide a natural language parser. A better representation for the sentence hypotheses is a word graph or lattice of word candidates which contains information on the approximate beginning and end point of each word's utterance to temporally relate the word candidates. We have conducted a simple experiment which demonstrates the compactness of a word graph. For this experiment, we selected three sets of N-best sentence hypotheses for three different types of utterances: a command, a yes-no question, and a wh-question. The list of the N-best sentences was converted to a word graph in which the duration of the node was determined by maintaining a syllable count through the utterance. The size of the constructed word graphs is compared with the number of words in the lists of N-best sentences (Ss) in table 1. The word graphs provided an 83% reduction in storage.

Sentence Type	Number Ss	Number Words	Distinct Words	Number Nodes	Words in Graph
Command	11	41	7	6	9
Yes-No-Q	20	129	17	11	18
Wh-Q	20	133	19	11	19

Table 1. N-best sentences versus a word graph.

Even though a word graph is a compact representation for the output of a speech recognition system, current systems do not provide this type of representation. However, parsers that can process the graph representation will more efficiently process all sentence hypotheses. Tomita [12] has developed an LR parsing algorithm capable of processing a word graph, though it does not use the graph directly in the parsing algorithm. On the other hand, we have developed a natural language framework based on Maruyama's constraint dependency grammar (CDG) [5, 6, 7] which allows us to process a word graph of sentence hypotheses directly.

In this paper, we will describe how a CDG parser performs the syntactic analysis of a single sentence and then describe the modifications required to process word graphs.

2 Constraint Dependency Grammars

To develop a syntactic analysis for a sentence using CDG, a constraint network (CN) of word nodes is constructed. Associated with each node is its position and a set of *roles*, which indicate the various functions the word fills in a sentence. Though two roles are required to write a grammar at least as expressive as a CFG [5], our examples depict a single role to simplify the discussion. The role described in the examples is the *governor* role, which represents the function 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 when it modifies the head noun.

Each role is initially assigned all *role values* allowed by the word's lexical category, where a role value consists of a label (the function the word can serve, e.g., SUBJ) and a modifiee (the number corresponding to the position of the word which it modifies, or nil). 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 modifiees or words) for each of the n words in the sentence, giving $O(n^2)$ role values altogether and requiring $O(n^2)$ time to generate. Figure 1 shows the initialization of the role values for the sentence *A fish eats.*

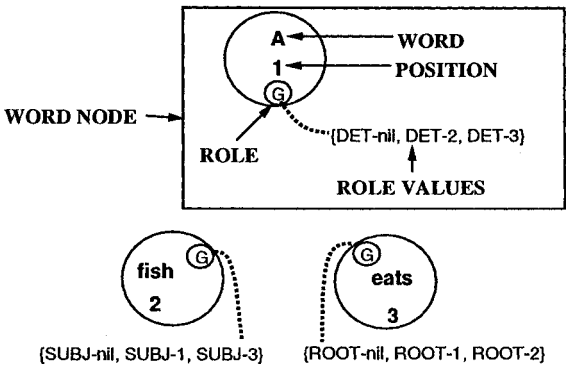

Figure 1. The word nodes for *A fish eats.*

Once the word nodes are constructed, constraints are applied to the role values to eliminate the ungrammatical ones. 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)

10.21437/ICSLP.1992-71

are applied to each role value to eliminate ungrammatical role values from the roles. For example, the following unary constraint eliminates all of the role values for *eats* except for ROOT-nil:

```
;; A verb has the governor label of ROOT and
;; a modifiee of nil.
(if (eq (category x) verb)
    (and (eq (label x) ROOT)
         (eq (modifiee x) nil)))
```

To apply this constraint to the network in figure 1, each role value for every role is examined to ensure that it obeys the constraint. A role value violates a constraint if and only if 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. Because a unary constraint can be tested against one role value in constant time and there are $O(n^2)$ role values to check, the time to apply a single unary constraint is $O(n^2)$. Initially, many unary constraints are applied to reduce the number of legal role values, requiring $O(k_u * n^2)$ time, where k_u represents the constant number of unary constraints. Additional unary constraints are applied to the network in figure 1 to eliminate the role values DET-nil, SUBJ-nil, and SUBJ-1 (e.g., a SUBJ must modify a verb and a DET must modify a noun).

Next, the CN is prepared for the propagation of *binary constraints*, which contain two variables and determine which pairs of role values can legally coexist. To keep track of pairs of role values, *arcs* connect the roles associated with each node to all other roles in the network. Each of the arcs has associated with it an *arc matrix*, whose row and column indices are the role values associated with the 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 either one or the other role value can exist, but not simultaneously). Initially, all entries in the matrices are set to **1**, indicating that there is nothing about one word's function which prohibits another word's right to have a certain function in the sentence. 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. 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 2 shows the network after the propagation of the following binary constraint:

```
;; A DET (determiner) is governed by a head noun
;; with the label of SUBJ (subject), OBJ (direct
;; object), IOBJ (indirect object), or PP_OBJ
;; (object of a preposition).
(if (and (equal (label x) DET)
         (equal (modifiee x) (position y)))
    (or (equal (label y) SUBJ)
        (equal (label y) OBJ)
        (equal (label y) IOBJ)
        (equal (label y) PP_OBJ)))
```

Since it is applied to $O(n^4)$ pairs of role values, the time to propagate the constraint is $O(n^4)$, and the time required to propagate k_b binary constraints is $O(k_b * n^4)$.

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. The illegal role values can be eliminated by *filtering* the CN. In filtering, a role value is removed from its role and from the row or column it indexes for each matrix associated with the arcs emanating from the role. For example, the role value DET-3 in figure 2 can be eliminated from the role for the word *a* and the rows indexed by those values can also be eliminated from the matrices on the arcs emanating from that role, resulting in an unambiguous parse for the sentence. The remaining role values form a parse graph for the sentence. A single application of filtering may be insufficient to eliminate illegal role values since the elimination of a role value from one role could lead to the elimination of a role value from another role. Filtering continues until there are no role values indexing matrix rows or columns containing only zeros, requiring $O(n^4)$ time (see [5]).

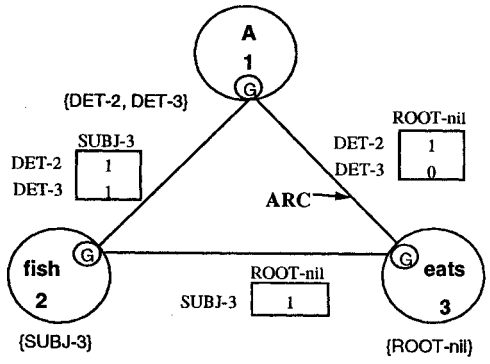

Figure 2. The CN for *A fish eats* after binary constraint propagation.

A CDG parser has several advantages over traditional CFG parsers. The set of languages which can be expressed by CDG is a superset of the context-free languages (e.g., Maruyama [5, 6] constructed a CDG grammar capable of parsing ww, where w is an arbitrary string of terminal symbols). CDG also allows the addition of a contextual dimension by applying different sets of constraints in different situations. This flexibility is an advantage of CDG over traditional CFG parsers, which use a single set of rules to parse all sentences. CDG provides the grammar designer with the flexibility to create grammars for free-order languages like Latin or to add the order constraints necessary to parse languages like English. Additionally, because CDG uses constraints instead of production rules, it is a simple matter to add exceptions without increasing the size of the grammar.

On the other hand, CDG has a slower serial running time than a CFG parser ($O(n^4)$ compared to $O(n^3)$, where n is the number of words in a sentence). However, we have devised a parallelization for the CDG parser [1] which uses $O(n^4)$ processors to parse in $O(k)$ time for a CRCW P-RAM model (Common Read, Common Write Parallel Random Access Machine), where n is the number of words in the sentence and k, the number of constraints, is a grammatical constant. Furthermore, this algorithm has been simulated on the MasPar MP-1, which uses the special features of the machine and $O(n^4)$ processors to obtain an $O(k + log(n))$ running time. CFG parsing algorithms have been parallelized [3]; however, to achieve sub-linear parse times has required $O(n^6)$ processors [8].

3 Serial CDG Parsing of Word Graphs

In this section, we describe how to augment a word graph to create and parse a Spoken Language Constraint Network (SLCN). Figure 3 depicts an SLCN derived from a word graph constructed for the sentence hypotheses: *A fish eat* and *Offices eats.* 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, but which could be the correct utterance. Each word node contains information on the beginning and end point of the utterance, represented as an integer tuple (b, e). The tuple is more expressive than the point scheme used for CNs and requires modification of the less-than and greater-than predicates used in constraints. Notice that word nodes contain a list of all word candidates with the same beginning and end points, and *edges* join word nodes that can be adjacent in a sen-

tence hypothesis (see figure 3).

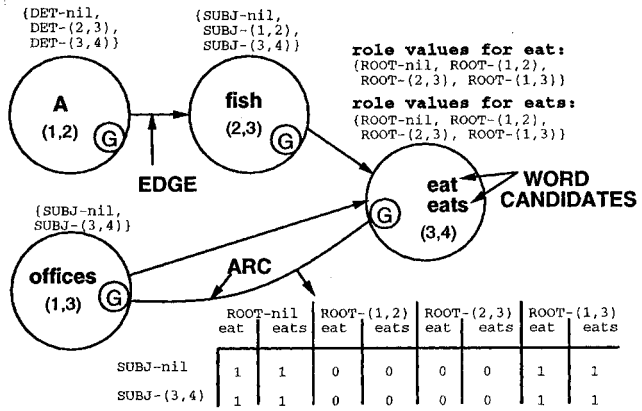

Figure 3. Example of a spoken language constraint network constructed from a word graph.

To parse an SLCN, each word candidate contained in a word node is assigned a set of role values for each role, requiring $O(n^2)$ time, where n is the number of word candidates in the graph. Unary constraints are applied to each of the role values in the network, as for CNs, requiring $O(k_u * n^2)$ time. The preparation of the SLCN for the propagation of binary constraints is similar to that for a CN, except that two roles are joined with an arc iff they can be members of the same sentence hypothesis (i.e., they are connected by a path of directed edges). For example, there should be an arc between the roles for *a* and *fish* in figure 3, but not between the roles for *a* and *offices*. Additionally, any of the matrix entries indexed by role values pointing to words not contained in the sentence hypothesis supporting an arc are automatically set to zero. In the SLCN of figure 3, the role values ROOT-(1,2) and ROOT-(2,3) cannot coexist with the role values for *offices* since they support another set of sentence hypotheses. The time required to prepare an SLCN for the propagation of binary constraints is $O(n^4)$. Like a CN, binary constraints are applied to pairs of role values in an SLCN, requiring $O(k_b * n^4)$ time.

Filtering in an SLCN is complicated by the fact that the limitation of one word's function in one sentence hypothesis should not necessarily limit that word's function in another sentence. A role value cannot be removed from a role in an SLCN until it is eliminated from the matrices of all arcs incident to that role. This is most easily illustrated by considering the arc matrix depicted in figure 3. Despite the fact that ROOT-(1,2) and ROOT-(2,3) are not supported by the role values for *offices*, neither should be ruled out until they are eliminated for the other sentence hypotheses. The filtering algorithm for an SLCN requires the introduction of a new notation for specifying the conditions under which a role value should be eliminated.

The following notational conventions are used to describe the algorithm. The capital letters **A**, **B**, **X**, and **Y** represent roles and the letter **r** represents a role value. Also, **arc_matrix(A,B)** represents the arc matrix for the arc connecting **A** to **B**. The term **connected-roles(B)** is the set of all roles which are connected to **B** with arcs and **supported-role-values(arc_matrix(A,B), B)** is a function which returns the list of role values corresponding to the role **B** which are supported by the **arc_matrix(A,B)** (i.e., the indexed row or column contains at least one **1**).

Suppose role **A** is connected by arcs to roles **B** and **X**. If **arc_matrix(A,B)** does not support a role value **r** associated with role **A** (i.e., **r** $\notin$ **supported-role-values(arc_matrix(A,B), A)**), then how can we determine whether **arc_matrix(A,X)** should continue to support **r** as a role value? The algorithm should eliminate **r** from **arc_matrix(A,X)** iff **X** is a member of *every sentence* that contains **B**; otherwise, it should not be eliminated. Note that **X** is a member of every sentence containing **B** in case 1 of figure 4, hence when **r** is eliminated from **arc_matrix(A,B)**, it should be eliminated from **arc_matrix(A,X)** and from the role **A**.

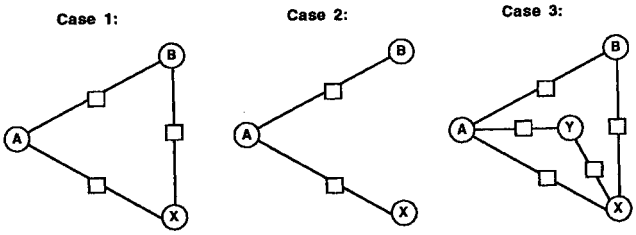

Figure 4. Filtering Cases for an SLCN.

The conditions in which the role value should be maintained are depicted in cases 2 and 3 in figure 4 and are enumerated below:

1. If **X** is not a member of any of the same sentences as **B** as shown in case 2 of figure 4 (i.e., **X** $\notin$ **connected-roles(B)**) then the role value **r** should remain supported by **arc_matrix(A,X)**.

2. Even if **X** is a member of some of the same sentences as **B** (i.e., **X** $\in$ **connected-roles(B)**), the role **r** should not be eliminated from **arc_matrix(A,X)** if **X** is contained in at least one other sentence not also containing **B**, as shown in case 3. Such a sentence exists only when there exists a role **Y** connected to roles **A** and **X**, but not to role **B**: $\exists$**Y** (**Y** $\in$ **connected-roles(A)**) $\wedge$ (**Y** $\neq$ **B**) $\wedge$ (**Y** $\in$ **connected-roles(X)**) $\wedge$ (**Y** $\notin$ **connected-roles(B)**) $\wedge$ (**r** $\in$ **supported-role-values(arc_matrix(A,X), A)**).

Before filtering an SLCN, a preprocessing step is performed to set up equivalence classes of arcs incident to each role, requiring $O(n^4)$ time. If a role value is eliminable from one arc in an equivalence class, it is eliminable from all of them. Filtering of an SLCN, like a CN, requires that each role value be examined to determine whether it is disallowed by some arc matrix (i.e., the row or column indexed by the role value contains only **0**s). However, if a matrix disallows a role value, then instead of that role value being automatically eliminated from the role and all of the incident arc matrices, the equivalence classes are used to determine which of the arc matrices should eliminate that role value. The role value is eliminated from the role iff it is disallowed by all arcs incident to that role. Filtering continues until there are no more role values to eliminate, requiring $O(n^4)$ time.

In an SLCN, if all of the role values in a role for a particular word candidate are eliminated, then that word candidate is removed from the list of supported words. If all of the word candidates for a word node are eliminated, then the word node is also eliminated along with all of the arcs and edges attached to that node. Furthermore, word nodes which are no longer members of a legal sentence hypothesis (because there exists no path of edges between the beginning and end of the sentence going through that node) are also eliminated, requiring up to $O(n)$ time.

The graphs created for the experiment in section 1 were converted to SLCNs and parsed using a grammar with three roles, 80 unary constraints, and 190 binary constraints. More grammatical sentences were parsed in the SLCN than were available in the original sets of sentences; however, all of the additional parses had similar meanings to at least one of the original grammatical N-best sentences.

Sentence Type	Number Grammatical Ss in N-best	Number Grammatical Ss in SLCN
Command	11	15
Yes-No-Q	8	12
Wh-Q	7	16

Table 2. N-best versus word graph sentence parses.

For example, consider the SLCN depicted in figure 5 and produced from the following N-best sentences:

```
1. Clear all windows.
2. Clear windows.
3. Clear all the windows.
4. Get all windows.
5. Give all windows.
6. Clear all of the windows.
7. Clear the windows.
8. Get all the windows.
9. Give all the windows.
10. Get all of the windows.
11. Give all of the windows.
```

The SLCN in figure 5 contains three verbs: *clear*, *give*, and *get*. Each is the main verb in 5 minor parse variations contained in the SLCN. The X-windows interface to the parser allows the user to view the role values for each word candidate's roles. The word *windows* over the interval (3,5) has one role value for each of its three roles, which can be viewed by clicking on it in the word node. This interface also allows the user to view the matrices stored on each of the arcs in the network.

The SLCN constraint parsing algorithm is rather slow, requiring $O(n^4)$ time to parse an SLCN with n word candidates. However, using a CRCW P-RAM model, an SLCN can be parsed in $O(k+n)$ time with $O(n^4)$ processors [2]. The extra n term is caused by the fact that when one word node is eliminated, any word nodes that are no longer members of a legal sentence hypothesis must also be eliminated from the network. If all of the word candidates were eliminated, this would require $O(n)$ time.

We are currently implementing the SLCN parsing algorithm on the MasPar MP-1. Because of the power of the MasPar and the parallel nature of the algorithm, we will not have to sacrifice the flexibility and expressivity of CDG grammars to improve performance when doing natural language parsing in a speech understanding system. We are also developing prosodic and semantic constraints to demonstrate how easily additional knowledge sources can be incorporated into CDG parsers.

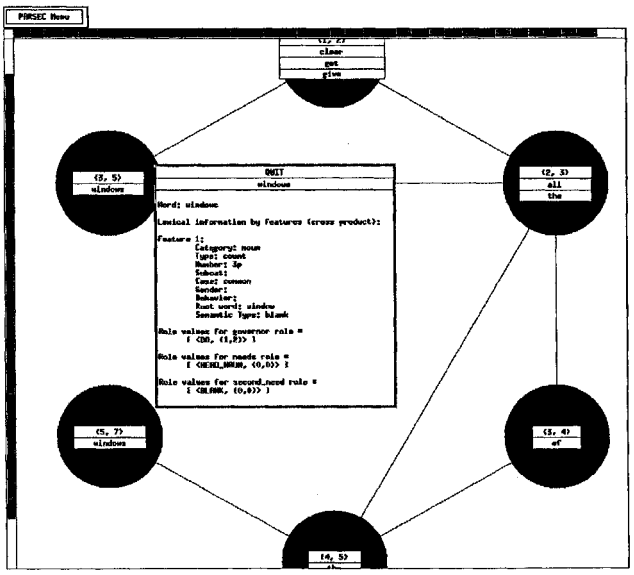

Figure 5. The X-windows interface to the SLCN for the N-best commands.

4 Acknowledgements

This work was supported in part by Purdue Researh 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] R. Helzerman and M.P. Harper. Log time parsing on the MasPar MP-1. In *Proceedings of the Sixth International Conference on Parallel Processing*, August 1992.

[2] 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.

[3] S.R. Kosaraju. Speed of recognition of context-free languages by array automata. *SIAM Journal of Computing*, 4(3) :331–340, September 1975.

[4] K. F. Lee, H. W. Hon, and R. Reddy. An overview of the SPHINX speech recognition system. In *IEEE Transactions on Acoustic, Speech, Signal Processing*, pages 35–45, January 1990.

[5] H. Maruyama. Constraint dependency grammar. Technical Report #RT0044, IBM, Tokyo, Japan, 1990.

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

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

[8] W. Ruzzo. Tree-size bounded alternation. *Journal of Computers and System Sciences*, 21:218–235, 1980.

[9] R. Schwartz and S. Austin. A comparison of several approximate algorithms for finding multiple N-best sentence hypotheses. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, May 1991.

[10] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul. Context-dependent modeling for acoustic-phonetic recognition of continuous speech. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, March 1985.

[11] 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.

[12] M. Tomita. An efficient word lattice parsing algorithm for continuous speech recognition. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, April 1986.