Syntactic Graphs and Constraint Satisfaction

Jeff Martin
Department of Linguistics, University of Maryland
College Park, MD 20742
jeffmar@umiacs.umd.edu

In this paper I will consider parsing as a discrete combinatorial problem which consists in constructing a labeled graph that satisfies a set of linguistic constraints. I will identify some properties of linguistic constraints which allow this problem to be solved efficiently using constraint satisfaction algorithms. I then describe briefly a modular parsing algorithm which constructs a syntactic graph using a set of generative operations and applies a filtering algorithm to eliminate inconsistent nodes and edges.

The model of grammar I will assume is not a monolithic rule system, but instead decomposes grammatical problems into multiple constraints, each describing a certain dimension of linguistic knowledge. The grammar is partitioned into *operations* and *constraints*. Some of these are given in (1); note that many constraints, including linear precedence, are not discussed here. I assume also that the grammar specifies a lexicon, which is a list of complex categories or *attribute-value structures* (Johnson 1988), along with a set of partial functions which define the possible categories of the grammar.

(1) Operations	Constraints
PROJECT-X	CASEMARK(X,Y)
ADJOIN-X	THETAMARK(X,Y)
MOVE-X	AGREE(X,Y)
INDEX-X	ANTECEDE(X,Y)

This cluster of modules incorporates operations and constraints from both GB theory (Chomsky 1981) and TAG (Johsi 1985). PROJECT-X is a category-neutral X-bar grammar consisting of three context-free metarules which yield a small set of unordered elementary trees. ADJOIN-X, which consists of a single adjunction schema, is a restricted type of tree adjunction which takes two trees and adjoins one to a projection of the head of the other. The combined schema are given in (2):

(2) X2 = { X1, Y2}	specifier axiom
X1 = { X0, Y2}	complement axiom
Xn = ω (a lexical category)	labeling axiom
Xn = {Xn, Yn}	adjunction axiom

MOVE-X constructs chains which link gaps to antecedents, while INDEX-X assigns indices to nodes from the set of natural numbers. In the parsing model to be discussed below, these make up the four basic operations of a nondeterministic automaton that generates sets of cantidate structures. Although these sets are finite, their size is not bounded above by a polynomial function in the size of the input. I showed in Martin(1989) that if X-bar and adjunction rules together allow four attachment levels, then the number of possible (unordered) trees formed by unconstrained application of these rules to a string of n terminals is $O(4^n)$. Also, Fong(1989) has shown that the number of distinct indexings for n noun phrases is $b_n = \Sigma_{m=1}^{n} \{{n \atop m}\}$, whose closed form solution is exponential. Unconstrained use of these operations therefore results in massive overgeneration, caused by the fact that they encode only a fragment of the knowledge in a grammar.

Unlike operations, the constraints in (1) crucially depend on the attributes of lexical items and nonterminal nodes. Three key properties of the constraints can be exploited to achieve an efficient filtering algorithm:

(i) they apply in local government configurations
(ii) they depend on node attributes whose domain of values is small
(iii) they are binary

For example, agreement holds between a phrase YP and a head Xo if and only if YP governs Xo, and YP and Xo share a designated agreement vector, such as [αperson, βnumber]; case marking holds between a head Xo and a phrase YP if and only if Xo governs YP, and Xo and YP share a designated case feature; and so forth. Lebeaux (1989) argues that only closed classes of features can enter into government relations. Unlike open lexical classes such as (3a), it is feasible to list the members of closed classes extensionally, for example the case features in (3b):

(3)a. Verb : {eat, sing, cry, ... }
b. Case : {Nom,Acc,Dat,Gen}

Constraints express the different types of attribute dependency which may hold between a governor and a governed node in a government domain. Each constraint can be represented as a binary predicate P(X,Y) which yields True if and only if a designated subset of attributes do not have distinct values in the categories X and Y. We may picture such predicates as specifying a path which must be *unifiable* in the directed acyclic graphs representing the categories X and Y.

Before presenting the outline of a parsing algorithm incorporating such constraints, it is necessary to introduce the notion of *boolean constraint satisfaction*

problem (BCSP) as defined in Mackworth (1987). Given a finite set of variables $\{V_1,V_2,...,V_n\}$ with associated domains $\{D_1,D_2,...,D_n\}$, constraint relations are stated on certain subsets of the variables; the constraints denote subsets of the cartesian product of the domains of those variables. The solution set is the largest subset of the cartesian product $D_1 \times D_2 \times ... \times D_n$ such that each n-tuple in that set satisfies all the constraints. Binary CSP's can be represented as a graph by associating a pair (V_i,D_i) with each node. An edge between nodes i and j denotes a binary constraint P_{ij} between the corresponding variables, while loops at a node i denote unary constraints P_i which restrict the domain of the node. Consistency is defined as follows:

(4) Node i is consistent iff $\forall x[x \in D_i] \Rightarrow P_i(x)$.
Arc i,j is consistent iff $\forall x[x \in D_i] \Rightarrow \exists y[y \in D_j \wedge P_{ij}(x,y)]$.
A path of length 2 from node i through node m to node j is consistent iff
$\forall x \forall z[P_{ij}(x,z)] \Rightarrow \exists y[y \in D_m \wedge P_{im}(x,y) \wedge P_{mj}(y,z)]$.

A network is node, arc, and path consistent iff all its nodes, arcs and paths are consistent. Path consistency can be generalized to paths of arbitrary length.

The parsing algorithm tries to find a consistent labeling for a syntactic graph representing the set of all syntactic analyses of an input string (see Seo & Simmons 1989 for a similar packed representation). The graph is constructed from left to right by the operations Project-X, Adjoin-X, Move-X and Index-X, which generate new nodes and arcs. In this scheme, overgeneration does not result in an abundance of parallel structures, but rather in the presence of superfluous nodes and arcs in a single graph. Each new node and arc generated is associated with a set of constraints; these associations are defined statically by the grammar. For example, complement arcs are associated with thetamarking constraints, specifier arcs are associated with agreement constraints, and indexing arcs are associated with coreference constraints. On each cycle the parser attempts to connect two consistently labeled subgraphs G1 and G2, where G1 represents the analyses of a leftmost portion of the input string, and G2 represents the analyses of the rightmost substring under consideration. The parse cycle contains three basic steps:

(a) select an operation
(b) apply the operation to graphs G_1 and G_2, yielding G_3
(c) apply node, arc and path consistency to the extended graph G_3.

Step (c) deletes inconsistent values from the domain at a node; also, if a node or arc is inconsistent, it is deleted. Note that nodes in syntactic graphs are labeled by linguistic categories which may contain many attribute-value pairs. Thus, a node typically represents not one but a set of variables whose values are relevant to the constraint predicates. The properties of locality and finite domains mentioned above turn out to be useful in the filtering step. Locality guarantees that the algorithm need only apply in a government domain. Therefore, it is not necessary to make the entire graph consistent after each extension, but only the largest subgraph which is a government domain and contains the nodes and edges most recently connected. The fact that the domains of attributes have a limited range is useful when the value of an attribute is unknown or ambiguous. In such cases, the number of possible solutions obtained by choosing an exact value for the attribute is small.

In this paper I have sketched the design of a parsing algorithm which makes direct use of a modular system of grammatical principles. The problem of overgeneration is solved by performing a limited amount of local computation after each generation step. This approach is quite different from one which preprocesses the grammar by folding together grammatical rules and constraints off-line. While this latter approach can achieve an a priori pruning of the search space by eliminating overgeneration entirely, it may do so at the cost of an explosion in grammar size.

References

Chomsky, N. (1981) *Lectures on Government and Binding*. Foris, Dordrecht.

Fong, S. (1990) "Free Indexation: Combinatorial Analysis and a Compositional Algorithm. *Proceedings of the ACL 1990.*

Johnson, M. (1988) *Attribute-Value Logic and the Theory of Grammar*. CSLI Lecture Notes Series, Chicago University Press.

Johsi, A. (1985) "Tree Adjoining Grammars," In D. Dowty, L. Karttunen & A. Zwicky (eds.), *Natural Language Processing*. Cambridge U. Press, Cambridge, England.

Lebeaux, D. (1989) *Language Acquisition and the Form of Grammar*. Doctoral dissertation, U. of Massachusetts, Amherst, Mass.

Mackworth, A. (1987) "Constraint Satisfaction," In: S Shapiro (ed.), *Encyclopedia of Artificial Intelligence*, Wiley, New York.

Mackworth, A. (1977) "Consistency in networks of relations," *Artif.Intell.* 8(1), 99-118.

Martin, J. (1989) "Complexity of Decision Problems in GB Theory," ms., U. of Maryland.

Seo, J. & R. Simmons (1989). "Syntactic Graphs: A Representation for the Union of All Ambiguous Parse Trees," *Computational Linguistics* 15:1.