Parsing Parallel Grammatical Representations

Derrick Higgins
Department of Linguistics
University of Chicago
1050 East 59th Street
Chicago, IL 60626
dchiggin@midway.uchicago.edu

Abstract

Traditional accounts of quantifier scope employ qualitative constraints or rules to account for scoping preferences. This paper outlines a feature-based parsing algorithm for a grammar with multiple simultaneous levels of representation, one of which corresponds to a partial ordering among quantifiers according to scope. The optimal such ordering (as well as the ranking of other orderings) is determined in this grammar not by absolute constraints, but by stochastic heuristics based on the degree of alignment among the representational levels. A Prolog implementation is described and its accuracy is compared with that of other accounts.

1 Introduction

It has long been recognized that the possibility and preference rankings of scope readings depend to a great degree on the position of scope-taking elements in the surface string (Chomsky, 1975; Hobbs and Shieber, 1987). Yet most traditional accounts of semantic scopal phenomena in natural language have not directly tied these two factors together. Instead, they allow only certain derivations to link the surface structure of a sentence with the representational level at which scope relations are determined, place constraints upon the semantic feature-passing mechanism, or otherwise emulate a constraint which requires some degree of congruence between the surface syntax of a sentence and its preferred scope reading(s).

A simpler and more direct approach is suggested by constraint-based, multistratal theories of grammar (Grimshaw, 1997; Jackendoff, 1997; Sadock, 1991; Van Valin, 1993). In these models, it is possible to posit multiple representational levels for a sentence without according ontological primacy to any one of them, as in all varieties of transformational grammar. This allows constraints to be formulated which place limits on structural discrepancies between levels, yet need not be assimilated into an overriding derivational mechanism.

This paper will examine the model of one of these theories, Autolexical Grammar (Sadock, 1991; Sadock, 1996; Schiller et al., 1996), as it is implemented in a computational scope generator and critic. This left-corner chart parser generates surface syntactic structures for each sentence (as the only level of syntactic representation), as well as Function-Argument semantic structures and Quantifier/Operator-Scope structures. These latter two structures together determine the semantic interpretation of a sentence.

It will be shown that this model is both categorical enough to handle standard generalizations about quantifier scope, such as bans on extraction from certain domains, and fuzzy enough to present reasonable preference rankings among scopings and account for lexical differences in quantifier strength (Hobbs and Shieber, 1987; Moran, 1988).

2 A Multidimensional Approach to Quantifier Scoping

2.1 The Autolexical Model

The framework of Autolexical Grammar treats a language as the intersection of numerous independent CF-PSGs, or *hierarchies*, each of which corresponds to a specific structural or functional aspect of the language. Semantic, syntactic, morphological, discourse-functional and many other hierarchies have been introduced in the literature, but this project focuses on the interactions among only three major hierarchies: Surface Syntax, Function-Argument Structure,

and Operator Scope Structure.

The surface syntactic hierarchy is a feature-based grammar expressing those generalizations about a sentence which are most clearly syntactic in nature, such as agreement, case, and syntactic valency. The function-argument hierarchy expresses that (formal) semantic information about a sentence which does not involve scope resolution, e.g., semantic valency and association of referential terms with argument positions, as in Park (1995). The operator scope hierarchy, naturally, imposes a scope ordering on the quantifiers and operators found in the expression. Two other, minor hierarchies are employed in this implementation. The linear ordering of words in the surface string is treated as a hierarchy, and a lexical hierarchy is introduced in order to express the differing lexical "strength" of quantifiers.

Each hierarchy can be represented as a tree in which the terminal nodes are not ordered with respect to one another. This implies that, for example, [John [saw Mary]] and [Mary [saw John]] will both be acceptable syntactic representations for the surface string *Mary saw John.* The optimal set of hierarchies for a string consists of the candidate hierarchies for each level of representation which together are most structurally congruous. The structural similarity between hierarchies is determined in Autolexical Grammar by means of an Alignment Constraint, which in the implementation described here counts the number of overlapping constituents in the two trees. Thus, while structures similar to [Mary [saw John]] and [John [saw Mary]] will both be acceptable as syntactic and function-argument structure representations, the alignment constraint will strongly favor a pairing in which both hierarchies share the same representation. Structural hierarchies are additionally evaluated by means of a Contiguity Constraint, which requires that the terminal nodes of each constituent of a hierarchy should be together in the surface string, or at least as close together as possible.

2.2 Quantifier Ordering Heuristics

The main constraints which this model places on the relative scope of quantifiers and operators are the alignment of the operator scope hierarchy with syntax, function-argument structure, and the lexical hierarchy of quantifier strength. The first of these constraints reflects "the principle that left-to-right order at the same syntactic level is preserved in the quantifier order"[1] and accounts for syntactic extraction restrictions. The second will favor operator scope structures in which scope-taking elements are raised as little as possible from their base argument positions. The last takes account of the scope preferences of individual quantifiers, such as the fact that *each* tends to have wider scope than all other quantifiers (Hobbs and Shieber, 1987; Moran, 1988).

As an example of the sort of syntactically-based restrictions on quantifier ordering which this model can implement, consider the generalization listed in Moran (1988), that "a quantifier cannot be raised across more than one major clause boundary." Because the approach pursued here already has a general constraint which penalizes candidate parses according to the degree of discrepancy between their syntax and scope hierarchies, we do not need to accord a privileged theoretical status to "major clause boundaries."

Figure 1 illustrates the approximate optimal structure accorded to the sentence *Some patients believe all doctors are competent* on the syntactic and scopal hierarchies, in which an extracted quantifier crosses one major clause boundary. It will be given a misalignment index of 4 (considering for the moment only the interaction of these two levels), because of the four overlapping constituents on the two hierarchies. This example would be misaligned only to degree 2 if the other quantifier order were chosen, and depending on the exact sentence type considered, an example with a scope-taking element crossing two major clause boundaries should be misaligned to about degree 8.

The fact that the difference between the primary and secondary scopings of this sentence is 2 degrees of alignment, while the difference between crossing one clause boundary and two clause boundaries is 4 degrees of alignment, corresponds with generally accepted assumptions about the acceptability of this example. While the reading in which the scope of quantifiers mirrors their order in surface structure is certainly preferred, the other ordering is possible as well. If the extraction crosses another clause

[1] Hobbs and Shieber (1987), p. 49

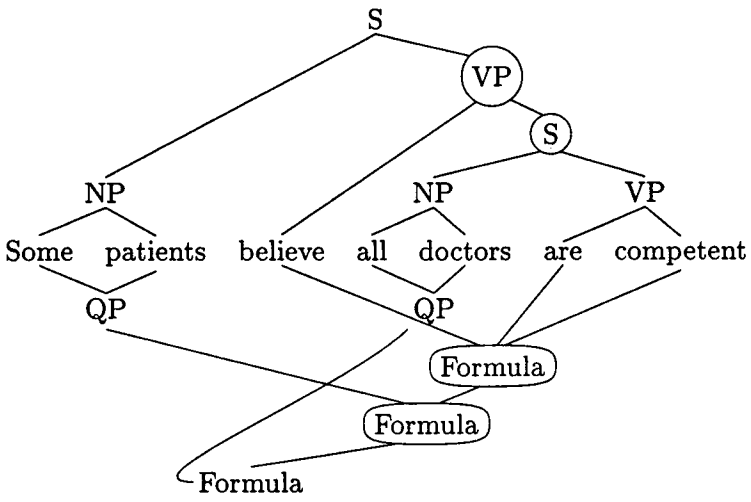

Figure 1: Illustration of the Alignment Constraint. The four highlighted nodes count against this combination of structures, because they overlap with constituents in the other tree.

boundary, however, as in *Some patients believe Mary thinks all doctors are competent*, the reversed scoping is considerably more unlikely.

2.3 Lexical Properties of Quantifiers

In addition to ranking the possible scopings of a sentence based on the surface syntactic positions of its quantifiers and operators, the parsing and alignment algorithm employed in this project takes into account the "strength" of different scope-taking elements. By introducing a lexical hierarchy of quantifier strength, in which those elements more likely to take wide scope are found higher in the tree, we are able to use the same mechanism of the alignment constraint to model the facts which other approaches treat with stipulative heuristics.

For example, in *Some patient paid each doctor*, the preferred reading is the one in which *each* takes wide scope, contrary to our expectations based on the generalization that the primary scoping tends to mirror surface syntactic order. An approach employing some variant of Cooper storage would have to account for this by assigning to each pair of quantifiers a likelihood that one will be raised past the other. In this case, it would be highly likely for *each* to be raised past *some*. The autolexical approach, however, allows us to achieve the same effect without introducing an additional device. Given a proper weighting of the result of aligning the scope hierarchy with this lexical hierarchy, it is a simple matter to settle on the correct candidates.

3 The Algorithm

3.1 Parsing Strategy

This implementation of the Autolexical account of quantifier scoping is written for SWI-Prolog, and inherits much of its feature-based grammatical formalism from the code listings of Gazdar and Mellish (1989), including `dagunify.pl`, by Bob Carpenter. The general strategy employed by the program is first to find all parses which each hierarchy's grammar permits for the string, and then to pass these lists of structures to functions which implement the alignment and contiguity constraints. These functions perform a pairwise evaluation of the agreement between structures, eventually converging on the optimal set of hierarchies.

The same parsing engine is used to generate structures for each of the major hierarchies contributing to the representation of a string. It is based on the left-corner parser of `pro_patr.pl` in Gazdar and Mellish (1989), attributed originally to Pereira and Shieber (1987). This parser has been extended to store intermediate results for lookup in a hash table.

At present, the parsing of each hierarchy is independent of that of the other hierarchies, but ultimately it would be preferable to allow, e.g., edges from the syntactic parse to contribute to

the function-argument parsing process. Such a development would allow us to express categorial prototypes in a natural way. For example, the proposition that "syntactic NPs tend to denote semantic arguments" could be modeled as a default rule for incorporating syntactic edges into a function-argument structure parse.

The "generate and test" mechanism employed here to maximize the congruity of representations on different levels is certainly somewhat inefficient. Some of the structures which it considers will be bizarre by all accounts. To a certain degree, this profligacy is held in check by heuristic cutoffs which exclude a combination from consideration as soon as it becomes apparent that is misaligned to an unacceptable degree. Ultimately, however, the solution may lie in some sort of parallel approach. A development of this program designed either for parallel Prolog or for a truly parallel architecture could effect a further restriction on the candidate set of representations by implementing constraints on parallel parsing processes, rather than (or in addition to) on the output of such processes.

3.2 Alignment

The alignment constraint (applied by the `align/3` predicate here) compares two trees (Prolog lists), returning the total number of overlapping constituents in both trees as a measure of their misalignent. Constituents are said to overlap if the sets of terminal nodes which they dominate intersect, but neither is a subset of the other.

The code fragment below provides a rough outline of the operation of this predicate. First, both trees being compared are "pruned" so that neither contains any terminal nodes not found in the other. The terminal elements of each of the tree's constituents are then recorded in lists. Once those constituents which occur in both trees are removed, the sum of the length of these two lists is the total number of overlapping constituents.

```
align(L1,L2,Num) :-

  flatten(L1,F1), flatten(L2,F2),
  union(F1,F2,AllTerms),
  intersection(F1,F2,GoodTerms),
  subtract(AllTerms,GoodTerms,BadTerms),

  % Delete constits without correlates
  rmbad(L1,BadTerms,Good1),
  rmbad(L2,BadTerms,Good2),

  % Get list of constits in each tree
  constits(Good1,CList1),
  constits(Good2,CList2),

  % Delete duplicates
  intersection(CList1,CList2,CList3),
  subtract(CList1,CList3,Final1),
  subtract(CList2,CList3,Final2),

  % Count mismatches
  length(Final1,Size1),
  length(Final2,Size2),
  Num is Size1 + Size2.
```

3.3 Contiguity

While the alignment constraint evaluates the similarity of two trees, the contiguity constraint (`contig/3` in this project) calculates the degree of fit between a hierarchy and a string (in this case, the surface string). The relevant measure of "goodness of fit" is taken here to be the minimal number of crossing branches the structure entails. It is true that this approach makes the contiguity constraint dependent on the particular grammatical rules of each representational level. However, since an Autolexical model does not attempt to handle syntax directly in the semantic representation, or morphology in the syntactic representation, there is no real danger of proliferating nonterminal nodes on any particular level.

The definition of the `contig` predicate is somewhat more complex than that for `align`, because it must find the *minimum* number of crossing branches in a structure. It works by maintaining a chart (based on the `contval` predicate) of the number of branches "covering" each constituent, as it works its way up the tree. The `contmin` predicate keeps track of the current lowest contiguity violation for the structure, so that worse alternatives can be abandoned as soon as they cross this threshold.

```
contig([],_,0).
contig(A,_,0) :-
  not(is_list(A)),
  !.
contig([A],Flat,Num) :-
```

```
  is_list(A),
  contig(A,Flat,Num),
  !.
contig([A,B],Flat,Num) :-
  contig(A,Flat,Num1),
  contig(B,Flat,Num2),
  contval(A,Left1,Right1,Num3),
  contval(B,Left2,Right2,Num4),
  Num0 is Num1 + Num2 + Num3 + Num4,
  forall(contmin(Min),
    (Num0 >= Min) *-> fail ; true),
  Num is Num0,
  forall(contval(X,L,R,N),
    (L > min(Left1,Left2),
      R < max(Right1,Right2)) *->
    (retract(contval(X,L,R,N)),
      asserta(contval(X,L,R,N+1)))
      ; true),
  asserta(
    contval([A,B],min(Left1,Left2),
    max(Right1,Right2),0)).
contig([B,A],Flat,Num) :-
  contig(A,Flat,Num1),
  contig(B,Flat,Num2),
  contval(A,Left1,Right1,Num3),
  contval(B,Left2,Right2,Num4),
  Num0 is Num1 + Num2 + Num3 + Num4,
  forall(contmin(Min),
    (Num0 >= Min) *-> fail ; true),
  Num is Num0,
  forall(contval(X,L,R,N),
    (L > min(Left1,Left2),
      R < max(Right1,Right2)) *->
    (retract(contval(X,L,R,N)),
      asserta(contval(X,L,R,N+1)))
      ; true),
  asserta(
    contval([A,B],min(Left1,Left2),
    max(Right1,Right2),0)).
```

4 Conclusion

Multistratal theories of grammar are not often chosen as guidelines for computational linguistics, because of performance and manageability concerns. This project, however, should at least demonstrate that even in a high-level language like Prolog a multistratal parsing model can be made to produce consistent results in a reasonable length of time.

Furthermore, the project described here does more than simply emulate the output of a standard, monostratal CF-PSG parser; it yields a preference ranking of readings for each string, rather than a single right answer. While the Autolexical model may not now be correct for applications in which speed is of primary concern, it has only begun to be implemented computationally, and any serious attempt at inferencing from natural language input will have to produce similar, graded output (Moran, 1988).

References

Noam Chomsky. 1975. Deep structure, surface structure, and semantic interpretation. In *Studies on Semantics in Generative Grammar*, pages 62–119. Mouton.

Gerald Gazdar and Chris Mellish. 1989. *Natural Language Processing in PROLOG.* Addison Wesley.

Jane Grimshaw. 1997. Projection, heads, and optimality. *Linguistic Inquiry*, 28(3):373–422.

Jerry R. Hobbs and Stuart M. Shieber. 1987. An algorithm for generating quantifier scopings. *Computational Linguistics*, 13:47–63.

Ray Jackendoff. 1997. *The Architecture of the Language Faculty.* Number 28 in Linguistic Inquiry Monographs. The MIT Press.

Douglas B. Moran. 1988. Quantifier scoping in the SRI core language engine. In *ACL Proceedings, 26th Annual Meeting*, pages 33–40.

Jong C. Park. 1995. Quantifier scope and constituency. In *ACL Proceedings, 33rd Annual Meeting.*

Fernando C.N. Pereira and Stuart M. Shieber. 1987. *Prolog and Natural-Language Analysis*, volume 10 of *CSLI Lecture Notes.* Center for the Study of Language and Information.

Jerrold M. Sadock. 1991. *Autolexical Syntax: a Theory of Parallel Grammatical Representations.* University of Chicago Press.

Jerrold M. Sadock. 1996. Reflexive reference in west greenlandic. *Contemporary Linguistics*, 1:137–160.

Eric Schiller, Elisa Steinberg, and Barbara Need, editors. 1996. *Autolexical Theory: Ideas and Methods.* Mouton de Gruyter.

Robert D. Van Valin, editor. 1993. *Advances in Role and Reference Grammar.* Number 82 in Current Issues in Linguistic Theory. John Benjamins Publishing Company.