The Computational Implementation of Principle-Based Parsers[1]

Sandiway Fong
Robert C. Berwick
Artificial Intelligence Laboratory,
Massachusetts Institute of Technology

Abstract

This paper addresses the issue of how to organize linguistic principles for efficient processing. Based on the general characterization of principles in terms of purely computational properties, the effects of principle-ordering on parser performance are investigated. A novel parser that exploits the possible variation in principle-ordering to dynamically re-order principles is described. Heuristics for minimizing the amount of unnecessary work performed during the parsing process are also discussed.

1 Introduction

Recently, there has been some interest in the implementation of grammatical theories based on the principles and parameters approach (Correa [3], Dorr [4], Johnson [5], Kolb & Thiersch [6], and Stabler [10]). In this framework, a fixed set of universal principles parameterized according to particular languages interact deductively to account for diverse linguistic phenomena. Much of the work to date has focused on the not inconsiderable task of formalizing such theories. The primary goal of this paper is to explore the computationally-relevant properties of this framework. In particular, we address the hitherto largely unexplored issue of how to organize linguistic principles for efficient processing. More specifically, this paper examines if, and how, a parser can re-order principles to avoid doing unnecessary work. Many important questions exist: for example, (1) What effect, if any, does principle-ordering have on the amount of work needed to parse a given sentence? (2) If the effect of principle-ordering is significant, then are some orderings much better than others? (3) If so, is it possible to predict (and explain) which ones these are?

By characterizing principles in terms of the purely computational notions of "filters" and "generators", we show how how principle-ordering can be utilized to minimize the amount of work performed in the course of parsing. Basically, some principles, like Move-α (a principle relating 'gaps' and 'fillers') and Free Indexing (a principle relating referential items) are "generators" in the sense that they build more hypothesized output structures than their inputs. Other principles, like the θ-Criterion which places restrictions on the assignment of thematic relations, the Case Filter which requires certain noun phrases to be

[1]The work of the first author is supported by an IBM Graduate Fellowship. R.C. Berwick is supported by NSF Grant DCR-85552543 under a Presidential Young Investigator's Award.

marked with abstract Case, and Binding Theory constraints, act as filters and weed-out ill-formed structures.

A novel, logic-based parser, the Principle-Ordering Parser (PO-PARSER), was built to investigate and demonstrate the effects of principle-ordering. The PO-PARSER was deliberately constructed in a highly-modular fashion to allow for maximum flexibility in exploring alternative orderings of principles. For instance, each principle is represented separately as an atomic parser operation. A structure is deemed to be well-formed only if it passes all parser operations. The scheduling of parser operations is controlled by a dynamic ordering mechanism that attempts to eliminate unnecessary work by eliminating ill-formed structures as quickly as possible. (For comparison purposes, the PO-PARSER also allows the user to turn off the dynamic ordering mechanism and to parse with a user-specified (fixed) sequence of operations.)

Although we are primarily interested in exploiting the (abstract) computational properties of principles to build more efficient parsers, the PO-PARSER is also designed to be capable of handling a reasonably wide variety of linguistic phenomena. The system faithfully implements most of the principles contained in Lasnik & Uriagereka's [7] textbook. That is, the parser makes the same grammaticality judgements and reports the same violations for ill-formed structures as the reference text. Some additional theory is also drawn from Chomsky [1] and [2]. Parser operations implement principles from Theta Theory, Case Theory, Binding Theory, Subjacency, the Empty Category Principle, movement at the level of Logical Form as well in overt syntax, and some Control Theory. This enables it to handle diverse phenomena including parasitic gaps constructions, strong crossover violations, passive, raising, and super-raising examples.

2 The Principle Ordering Problem

This section addresses the issue of how to organize linguistic principles in the PO-PARSER framework for efficient processing. More precisely, we discuss the problem of how to order the application of principles to minimize the amount of 'work' that the parser has to perform. We will explain why certain orderings may be better in this sense than others. We will also describe heuristics that the PO-PARSER employs in order to optimize the the ordering of its operations.

But first, is there a significant performance difference between various orderings? Alternatively, how important an issue is the principle ordering problem in parsing? An informal experiment was conducted using the PO-PARSER described in the previous section to provide some indication on the magnitude of the problem. Although we were unable to examine all the possible orderings, it turns out that order-of-magnitude variations in parsing times could be achieved merely by picking a few sample orderings.[2]

[2]The PO-PARSER has about twelve to sixteen parser operations. Given a set of one dozen operations, there are about 500 million different ways to order these operations. Fortunately, only about half a million of these are actually valid, due to logical dependencies between the various operations. However, this is still far too many to test exhaustively. Instead, only a few well-chosen orderings were tested on a number of sentences from the reference. The procedure

2.1 Explaining the Variation in Principle Ordering

The variation in parsing times for various principle orderings that we observed can be explained by assuming that overgeneration is the main problem, or bottleneck, for parsers such as the PO-PARSER. That is, in the course of parsing a single sentence, a parser will hypothesize many different structures. Most of these structures, the ill-formed ones in particular, will be accounted for by one or more linguistic filters. A sentence will be deemed acceptable if there exists one or more structures that satisfy every applicable filter. Note that even when parsing grammatical sentences, overgeneration will produce ill-formed structures that need to be ruled out. Given that our goal is to minimize the amount of work performed during the parsing process, we would expect a parse using an ordering that requires the parser to perform extra work compared with another ordering to be slower.

Overgeneration implies that we should order the linguistic filters to eliminate ill-formed structures as quickly as possible. For these structures, applying any parser operation other than one that rules it out may be considered as doing extra, or unnecessary, work (modulo any logical dependencies between principles).[3] However, in the case of a well-formed structure, principle ordering cannot improve parser performance. By definition, a well-formed structure is one that passes all relevant parser operations: Unlike the case of an ill-formed structure, applying one operation cannot possibly preclude having to apply another.

2.2 Optimal Orderings

Since some orderings perform better than others, a natural question to ask is: Does there exist a 'globally' optimal ordering? The existence of such an ordering would have important implications for the design of the control structure of any principle-based parser. The PO-PARSER has a novel 'dynamic' control structure in the sense that it tries to determine an ordering-efficient strategy for every structure generated. If such a globally optimal ordering could be found, then we can do away with the run-time overhead and parser machinery associated with calculating individual orderings. That is, we can build an ordering-efficient parser simply by 'hardwiring' the optimal ordering into its control structure. Unfortunately, no such ordering can exist.

The impossibility of the globally optimal ordering follows directly from the "eliminate unnecessary work" ethic. Computationally speaking, an optimal ordering is one that rules out ill-formed structures at the earliest possible opportunity. A *globally* optimal ordering would be one that always ruled out every

---

involved choosing a default sequence of operations and 'scrambling' the sequence by moving operations as far as possible from their original positions (modulo any logical dependencies between operations).

[3] In the PO-PARSER for example, the Case Filter operation which requires that all overt noun phrases have abstract Case assigned, is dependent on both the inherent and structural Case assignment operations. That is, in any valid ordering the filter must be preceded by both operations.

possible ill-formed structure without doing any unnecessary work. Consider the following three structures (taken from Lasnik's book):

(1) a. *John$_1$ is crucial $[_{CP}[_{IP}\ t_1$ to see this]]
  b. *$[_{NP}$John$_1$'s mother $][_{VP}$ likes himself$_1$]
  c. *John$_1$ seems that he$_1$ likes t_1

Example (1) violates the Empty Category Principle (ECP). Hence the optimal ordering must invoke the ECP operation before any other operation that it is not dependent on. On the other hand, example (1b) violates a Binding Theory principle, 'Condition A'. Hence, the optimal ordering must also invoke Condition A as early as possible. In particular, given that the two operations are independent, the optimal ordering must order Condition A before the ECP and vice-versa. Similarly, example (1c) demands that the 'Case Condition on Traces' operation must precede the other two operations. Hence a globally optimal ordering is impossible.

2.3 Heuristics for Principle Ordering

The principle-ordering problem can be viewed as a limited instance of the well-known conjunct ordering problem (Smith & Genesereth [9]). Given a set of conjuncts, we are interested in finding all solutions that satisfy all the conjuncts simultaneously. The parsing problem is then to find well-formed structures (i.e. solutions) that satisfy all the parser operations (i.e. conjuncts) simultaneously. Moreover, we are particularly interested in minimizing the cost of finding these structures by re-ordering the set of parser operations.

This section outlines some of the heuristics used by the PO-PARSER to determine the minimum cost ordering for a given structure. The PO-PARSER contains a dynamic ordering mechanism that attempts to compute a minimum cost ordering for every phrase ucture generated during the parsing process.[4] The mechanism can be subd ied into two distinct phases. First, we will describe how the dynamic ordering mechanism decides which principle is the most likely candidate for eliminating a given structure. Then, we will explain how it makes use of this information to re-order parser operation sequences to minimize the total work performed by the parser.

2.3.1 Predicting Failing Filters

Given any structure, the dynamic ordering mechanism attempts to satisfy the "eliminate unnececessary work" ethic by predicting a "failing" filter for that

[4] In their paper, Smith & Genesereth drew a distinction between "static" and "dynamic" ordering strategies. In static strategies, the conjuncts are first ordered, and then solved in the order presented. By contrast, in dynamic strategies the chosen ordering may be revised between solving individual conjuncts. Currently, the PO-PARSER employs a dynamic strategy. The ordering mechanism computes an ordering based on certain features of each structure to be processed. The ordering may be revised after certain operations (e.g. movement) that modify the structure in question.

structure. More precisely, it will try to predict the principle that a given structure violates on the basis of the simple structure cues. Since the ordering mechanism cannot know whether a structure is well-formed or not, it assumes that all structures are ill-formed and attempts to predict a failing filter for every structure. In order to minimize the amount of work involved, the types of cues that the dynamic ordering mechanism can test for are deliberately limited. Only inexpensive tests such as whether a category contains certain features (e.g. ±anaphoric, ±infinitival, or whether it is a trace or a non-argument) may be used. Any cues that may require significant computation, such as searching for an antecedent, are considered to be too expensive. Each structure cue is then associated with a list of possible failing filters. (Some examples of the mapping between cues and filters are shown below.) The system then chooses one of the possible failing filters based on this mapping.[5]

(2)

Structure cue	*Possible failing filters*
trace	Empty Category Principle, and Case Condition on traces
intransitive	Case Filter
passive	Theta Criterion Case Filter
non-argument	Theta Criterion
+anaphoric	Binding Theory Principle A
+pronominal	Binding Theory Principle B

The correspondence between each cue and the set of candidate filters may be systematically derived from the definitions of the relevant principles. For example, Principle A of the Binding Theory deals with the conditions under which antecedents for anaphoric items, such as "each other" and "himself", must appear. Hence, Principle A can only be a candidate failing filter for structures that contain an item with the +anaphoric feature. Other correspondences may be somewhat less direct: for example, the Case Filter merely states that all overt noun phrase must have abstract Case. Now, in the PO-PARSER the conditions under which a noun phrase may receive abstract Case are defined by two separate operations, namely, Inherent Case Assignment and Structural Case Assignment. It turns out that an instance where Structural Case Assignment will not assign Case is when a verb that normally assigns Case has passive morphology. Hence, the presence of a passive verb in a given structure may cause an overt noun phrase to fail to receive Case during Structural Case Assignment — which, in turn may cause the Case Filter to fail.[6]

---

[5] Obviously, there are many ways to implement such a selection procedure. Currently, the PO-PARSER uses a voting scheme based on the frequency of cues. The (unproven) underlying assumption is that the probability of a filter being a failing filter increases with the number of occurrences of its associated cues in a given structure. For example, the more traces there are in a structure, the more likely it is that one of them will violate some filter applicable to traces, such as the Empty Category Principle (ECP).

[6] It is possible to automate the process of finding structure cues simply by inspecting the closure of the definitions of each filter and all dependent operations. One method of deriving

The failing filter mechanism can been seen as an approximation to the Cheapest-first heuristic in conjunct ordering problems. It turns out that if the cheapest conjunct at any given point will reduce the search space rather than expand it, then it can be shown that the optimal ordering must contain that conjunct at that point. Obviously, a failing filter is a "cheapest" operation in the sense that it immediately eliminates one structure from the set of possible structures under consideration.

Although the dynamic ordering mechanism performs well in many of the test cases drawn from the reference text, it is by no means foolproof. There are also many cases where the prediction mechanism triggers an unprofitable re-ordering of the default order of operations. (We will present one example of this in the next section.) A more sophisticated prediction scheme, perhaps one based on more complex cues, could increase the accuracy of the ordering mechanism. However, we will argue that it is not cost-effective to do so. The basic reason is that, in general, there is no *simple* way to determine whether a given structure will violate a certain principle.[7] That is, as far as one can tell, it is difficult to produce a cheap (relative to the cost of the actual operation itself), but effective approximation to a filter operation. For example, in Binding Theory, it is difficult to determine if an anaphor and its antecedent satisfies the complex locality restrictions imposed by Principle A without actually doing some searching for a binder. Simplifying the locality restrictions is one way of reducing the cost of approximation, but the very absence of search is the main reason why the overhead of the present ordering mechanism is relatively small.[8] Hence, having more sophisticated cues may provide better approximations, but the tradeoff is that the prediction methods may be almost as expensive as performing the real operations themselves.

2.3.2 Logical Dependencies and Re-ordering

Given a candidate failing filter, the dynamic ordering mechanism has to schedule the sequence of parser operations so that the failing filter is performed as early

---

cues is to collect the negation of all conditions involving category features. For example, if an operation contains the condition "`not (Item has_feature intransitive)`", then we can take the presence of an intransitive item as a possible reason for failure of that operation. However, this approach has the potential problem of generating too many cues. Although, it may be relatively inexpensive to test each individual cue, a large number of cues will significantly increase the overhead of the ordering mechanism. Furthermore, it turns out that not all cues are equally useful in predicting failure filters. One solution may be to use "weights" to rank the predictive utility of each cue with respect to each filter. Then an adaptive algorithm could be used to "learn" the weighting values, in a manner reminiscent of Samuels [8]. The failure filter prediction process could then automatically eliminate testing for relatively unimportant cues. This approach is currently being investigated.

[7] If such a scheme can be found, then it can effectively replace the definition of the principle itself.

[8] We ignore the additional cost of re-ordering the sequence of operations once a failing filter has been predicted. The actual re-ordering can be made relatively inexpensive using various tricks. For example, it is possible to "cache" or compute (off-line) common cases of re-ordering a default sequence with respect to various failing filters, thus reducing the cost of re-ordering to that of a simple look-up.

as possible. Simply moving the failing filter to the front of the operations queue is not a workable approach for two reasons.

Firstly, simply fronting the failing filter may violate logical dependencies between various parser operations. For example, suppose the Case Filter was chosen to be the failing filter. To create the conditions under which the Case Filter can apply, both Case assignment operations, namely, Inherent Case Assignment and Structural Case Assignment, must be applied first. Hence, fronting the Case Filter will also be accompanied by the subsequent fronting of both assignment operations — unless, of course, they have already been applied to the structure in question.

Secondly, the failing filter approach does not take into account the behaviour of "generator" operations. A generator may be defined as any parser operation that always produces one output, and possibly more than one output, for each input. For example, the operations corresponding to $\bar{X}$ rules, Move-α, Free Indexing and LF Movement are the generators in the PO-PARSER. (Similarly, the operations that we have previously referred to as "filters" may be characterized as parser operations that, when given N structures as input, pass N and possibly fewer than N structures.) Due to logical dependencies, it may be necessary in some situations to invoke a generator operation before a failure filter can be applied. For example, the filter Principle A of the Binding Theory is logically dependent on the generator Free Indexing to generate the possible antecedents for the anaphors in a structure. Consider the possible binders for the anaphor "*himself*" in "*John thought that Bill saw himself*" as shown below:

(3) a. *John$_i$ thought that Bill$_j$ saw himself$_i$
b. John$_i$ thought that Bill$_j$ saw himself$_j$
c.*John$_i$ thought that Bill$_j$ saw himself$_k$

Only in example (3b), is the antecedent close enough to satisfy the locality restrictions imposed by Principle A. Note that Principle A had to be applied a total of three times in the above example in order to show that there is only one possible antecedent for "*himself*". This situation arises because of the general tendency of generators to overgenerate. But this characteristic behaviour of generators can greatly magnify the extra work that the parser does when the dynamic ordering mechanism picks the wrong failing filter. Consider the ill-formed structure "**John seems that he likes t*" (a violation of the principle that traces of noun phrase cannot receive Case.) If however, Principle B of the Binding Theory is predicted to be the failure filter (on the basis of the structure cue "*he*"), then Principle B will be applied repeatedly to the indexings generated by the Free Indexing operation. On the other hand, if the Case Condition on Traces operation was correctly predicted to be the failing filter, then Free Indexing need not be applied at all. The dynamic ordering mechanism of the PO-PARSER is designed to be sensitive to the potential problems caused by selecting a candidate failing filter that is logically dependent on many generators.[9]

[9] Obviously, there are many different ways to accomplish this. One method is to compute

2.4 Linguistic Filters and Determinism

In this section we describe how the characterization of parser operations in terms of filters and generators may be exploited further to improve the performance of the PO-PARSER for some operations. More precisely, we make use of certain computational properties of linguistic filters to improve the backtracking behaviour of the PO-PARSER. The behaviour of this optimization will turn out to complement that of the ordering selection procedure quite nicely. That is, the optimization is most effective in exactly those cases where the selection procedure is least effective.

We hypothesize that linguistic filters, such as the Case Filter, Binding Conditions, ECP, and so on, may be characterized as follows:

> (4) **Hypothesis**: Linguistic filters are side-effect free conditions on configurations

In terms of parser operations, this means that filters should never cause structure to be built or attempt to fill in feature slots.[10] Moreover, computationally speaking, the parser operations corresponding to linguistic filters should be deterministic. That is, any given structure should either fail a filter or just pass. A filter operation should never need to succeed more than once, simply because it is side-effect free.[11] By contrast, operations that we have characterized as generators, such as Move-α and Free Indexing, are not deterministic in this sense. That is, given a structure as input, they may produce one or more structures as output.

---

the "distance" of potential failure filters from the current state of the parser in terms of the number of generators yet to be applied. Then the failing filter will be chosen on the basis of some combination of structure cues and generator distance. Currently, the PO-PARSER uses a slightly different and cheaper scheme. The failure filter is chosen solely on the basis of structure cues. However, the fronting mechanism is restricted so that the chosen filter can only move a limited number of positions ahead of its original position. The original operation sequence is designed such that the distance of the filter from the front of the sequence is roughly proportional to the number of (outstanding) operations that the filter is dependent on.

[10] So far, we have not encountered any linguistic filters that require either structure building or feature assignment. Operations such as θ-role and Case assignment are not considered filters in the sense of the definition given in the previous section. In the PO-PARSER, these operations will never fail. However, definitions that involve some element of 'modality' are potentially problematic. For example, Chomsky's definition of an *accessible SUBJECT*, a definition relevant to the principles of Binding Theory, contains the following phrase "*... assignment to α of the index of β would not violate the (i-within-i) filter* $*[_{\gamma_i} \ldots \delta_i \ldots]$". A transparent implementation of such a definition would seem to require some manipulation of indices. However, Lasnik (p.58) points out that there exists an empirically indistinguishable version of *accessible SUBJECT* without the element of modality present in Chomsky's version.

[11] It turns out that there are situations where a filter operation (although side-effect free) could succeed more than once. For example, the linguistic filter known as the "Empty Category Principle" (ECP) implies that all traces must be "properly governed". A trace may satisfy proper government by being either "lexically governed" or "antecedent governed". Now consider the structure $[_{CP}$ *What*$_1$ *did you* $[_{VP}$ *read* $t_1]]$. The trace t_1 is both lexically governed (by the verb *read*) and antecedent governed (by its antecedent *what*). In the PO-PARSER the ECP operation can succeed twice for cases such as t_1 above.

Given the above hypothesis, we can cut down on the amount of work done by the PO-PARSER by modifying its behaviour for filter operations. Currently, the parser employs a backtracking model of computation. If a particular parser operation fails, then the default behaviour is to attempt to re-satisfy the operation that was called immediately before the failing operation. In this situation, the PO-PARSER will only attempt to re-satisfy the preceding operation if it happens to be a generator. When the preceding operation is a filter, then the parser will skip the filter and, instead, attempt to resatisfy the next most recent operation and so on.[12] For example, consider the following calling sequence:

$$\boxed{G_1} \cdots \longrightarrow \boxed{G_2} \longrightarrow \boxed{F_1} \longrightarrow \boxed{F_2} \longrightarrow \boxed{F_3}$$

Suppose that a structure generated by generator G_2 passes filters F_1 and F_2, but fails on filter F_3. None of the three filters could have been the cause of the failure by the side-effect free hypothesis. Hence, we can skip trying to resatisfy any of them and backtrack straight to G_2.

Note that this optimization is just a limited form of dependency-directed backtracking. Failures are traced directly to the last generator invoked, thereby skipping over any intervening filters as possible causes of failure. However, the backtracking behaviour is limited in the sense that the most recent generator may not be the cause of a failure. Consider the above example again. The failure of F_3 need not have been caused by G_2. Instead, it could have been caused by structure-building in another generator further back in the calling sequence, say G_1. But the parser will still try out all the other possibilities in G_2 first.

Consider a situation in which the principle selection procedure performs poorly. That is, for a particular ill-formed structure, the selection procedure will fail to immediately identify a filter that will rule out the structure. The advantages of the modified mechanism over the default backtrack scheme will be more pronounced in such situations — especially if the parser has to try several filters before finding a "failing" filter. By contrast, the behaviour of the modified mechanism will resemble that of the strict chronological scheme in situations where the selection procedure performs relatively well (i.e. when a true failing filter is fronted). In such cases, the advantages, if significant, will be small. (In an informal comparison between the two schemes using about eighty sentences from the reference text, only about half the test cases exhibited a noticeable decrease in parsing time.)

[12] This behaviour can be easily simulated using the 'cut' predicate in Prolog. We can route all calls to filter operations through a predicate that calls the filter and then cuts off all internal choice points. (For independent reasons, the PO-PARSER does not actually use this approach.)

3 Conclusions: The Utility of Principle-Ordering

From our informal experiments with the PO-PARSER, we have found that dynamic principle-ordering can provide a significant improvement over any fixed ordering. We have found that speed-ups varying from three- or four-fold to order-of-magnitude improvements are possible in many cases.[13]

The control structure of the PO-PARSER forces linguistic principles to be applied one at a time. Many other machine architectures are certainly possible. For example, we could take advantage of the independence of many principles and apply principles in parallel whenever possible. However, any improvement in parsing performance would come at the expense of violating the minimum (unnecessary) work ethic. Lazy evaluation of principles is yet another alternative. However, principle-ordering would still be an important consideration for efficient processing in this case. Finally, we should also consider principle-ordering from the viewpoint of scalability. The experience from building prototypes of the PO-PARSER suggests that as the level of sophistication of the parser increases (both in terms of the number and complexity of individual principles), the effect of principle-ordering also becomes more pronounced.

References

[1] Chomsky, N.A., *Lectures on Government and Binding: The Pisa Lectures.* 1981. Foris Publications.

[2] Chomsky, N.A., *Knowledge of Language: Its Nature, Origin, and Use.*" 1986. Prager.

[3] Correa, N., "Syntactic Analysis of English with respect to Government-Binding Grammar," Ph.D Dissertation, 1988. Syracuse University.

[4] Dorr, B.J., "UNITRAN: A Principle-Base Approach to Machine Translation," M.I.T. A.I. Technical Report No. 1000. 1987.

[5] Johnson, M., "Knowledge as Language," *ms.* M.I.T. Brain and Cognitive Sciences.

[6] Kolb, H.P., & C. Thiersch, "Levels and Empty Categories in a Principles and Parameters Approach to Parsing," *ms.* 1988. Tilburg University.

[7] Lasnik, H. & J. Uriagereka, *A Course in GB Syntax: Lectures on Binding and Empty Categories.* 1988. M.I.T. Press.

[8] Samuels, A.L., "Some Studies in Machine Learning Using the Game of Checkers. II — Recent Progress," *IBM Journal.* November 1967.

[9] Smith, D.E., & M.R. Genesereth, "Ordering Conjunctive Queries," *Artificial Intelligence* **26** (1985) 171–215.

[10] Stabler, E.P., Jr. "The Logical Approach to Syntax: Foundations, Specifications and Implementations of Theories of Government and Binding." *ms.* 1989. University of Western Ontario.

[13]Obviously, the speed-up obtained will depend on the number of principles present in the system and the degree of 'fine-tuning' of the failure filter selection criteria.