Design Requirements for Principle-Based Parsers as Flexible Research Tools

Sebastian Millies
Dept. of Computational Linguistics
University of Saarbrücken
e-mail: millies@coli.uni-sb.de

Abstract

One principal aim in the design of principle-based parsers is to mirror in the parser the structure of linguistic theory. This raises the issue of what processing model is appropriate to interacting principles. Parser control strategies are also important in modelling aspects of linguistic performance. At the same time, the rapid development in linguistic theory calls for a flexible architecture, in which it is possible to experiment with different versions of the grammar. This kind of flexibility, together with a large degree of independence between representation and control, will be necessary for bridging the gap between computational and theoretical linguistics by providing a research tool useful to both sides.

1 Introduction: Why PBP?

In this paper, we will discuss some of the decisions that will confront every implementor of a principle-based parsing system.[1] Existing PBP systems will be cited to illustrate the possible outcomes of different decisions at different points during the design process.[2] Accordingly, this paper is not a presentation in depth of any particular implementation, but rather an overview of procedural models of GB grammar. The discussion will be a general one, not going into unnecessary technical detail. In this context, we will restrict our attention to *modular* systems, which mirror the internal structure of linguistic theory in such a way that components of the parser correspond to subtheories of universal grammar.[3] Such

---

1 I will understand "principle-based parsing" to be synonymous with "GB-parsing". Both terms mean parsing based on theories of the principles-and-parameters type pioneered by Chomsky in [Chom81]. This paper will presuppose a basic familiarity with GB. For a short introduction, see [Sel85].

2 Among them, my own system is described in more detail in [Mil90].

3 Obviously, it is not enough to require of a principle-based parser that it somehow respect the principles of GB in constructing its output, for then any adequate parser for a natural language

systems (experimental ones, to be sure) have become somewhat fashionable lately, and it may be appropriate to ask whether they are just play-things, or whether they may evolve to aid either computational or theoretical linguists - or perhaps even both - in their research. It turns out that the interests both sides have in the project are closely related: Theoretical linguistics is conducted in an essentially informal manner, so theoretical linguists may profit from PBP if it succeeds in providing a way to easily formalize the theory (cf. [Sta89]) and test its predictions, thereby giving more rigour and a firmer empirical footing to theories of grammar.[4] Computational linguists, on the other hand, will profit from such systems if they obviate the need to reconstruct insights of GB syntax in a radically different framework, thereby shortening the development cycle for grammars used in NL systems. So both sides will essentially expect to be provided with a high-level GB programming language. In comparison to rule-based systems, one would expect principle-based systems based on this new implementation language to be less ad-hoc (because no ad-hoc rules can be added to cover special constructions), more portable and versatile (because they encode much more knowledge in a language independent way), and smaller (because principles generalize over constructions and languages, so there are fewer of them).[5] As an afterthought, let us add that principle-based systems may also yield a better psychological model of human sentence processing than rule-based ones.

2 Computational differences between rules and principles

In rule-based grammars, every rule builds a local tree. In principle-based grammars, many independent principles put their individual constraints on a piece (not necessarily local) of structure. So where does the structure come from? Obviously, there seems to have to be *some* sort of generator in the system. A popular approach is to include $\overline{X}$-theory as a rule-based component. This makes possible the exploitation of standard parsing techniques (e.g., [Fong91] uses LR(1)-parsing). However, as $\overline{X}$-theory defines the entire set of trees (binary trees, in most versions) as possible structures, this approach is likely to yield parsers which generate a lot of ill-formed structure, or even do not terminate for ungrammatical input. More importantly, it is also contrary to the spirit of GB. In modern grammatical theory, $\overline{X}$-theory is often not viewed as a proper module of grammar.

---

would qualify as principle-based, if only GB were correct. This criterion excludes, for example, the Marcus parser, which [BW84] have claimed to be principle-based.

4 Foremost in this respect is the work of Stabler.

5 This is one of the reasons why compiling a principle-based system out into a huge set of rules will not necessarily give one a more efficient parser.

Instead, phrase structure is seen as emerging from other grammatical relations, such as case- or θ-marking and the linear order of heads and complements. The extreme approach, tried in [Abn86], would try to recover these relations directly and dispense with phrase structure altogether. A more sensible approach would try to restrict the phrase structure building component by additional information, so that only locally well-formed trees can be built.[6] For example, Millies uses subcategorization information from the lexicon to do this ([Mil90]). As much of grammatical theory as possible should be enforced immediately on local structure. Several methods have been developed to locally compute seemingly global constraints like subjacency. The indexing schemes of Latecki ([Lat91]) are particularly suited to this purpose. It is as yet unclear whether there are "essentially global" principles that cannot be treated this way, and also cannot be replaced by a series of local computations.[7]

3 The relation of GB grammars to GB parsers

We consider *flexibility* of the foremost importance in PBP systems. Theoretical linguistics is rapidly changing, and every linguist has its own favorite version of it. There is little sense in "hardwiring" a particular version of GB into a parser. The computational advantages of flexibility are easy maintenance and updatability. There seem to be at least two necessary conditions for this kind of flexibility:

- *Directness:*
  There is a one-to-one mapping from the linguistic vocabulary to the parser vocabulary.

- *Faithfulness:*
  The logical structure of the theory is mirrored by the computational structure of the parser.

Directness and Faithfulness are generally considered to be among the defining properties of the notion of GB-parser (cf. footnote 2). What makes these conditions so difficult to implement is the fact that theoretical linguistics places no constraints on the kinds of principles it allows and the notions used in expressing them.[8] There is no well defined set of primitive notions in GB, indeed, finding

---

6 One technique to achieve this is by an interleaving strategy, discussed below.

7 A good candidate for such a principle is the ECP defined in terms of barriers. Although indexing methods have been used to implement the ECP (e.g. [Mil90]), this relies on goal freezing and may still lead to generation of intermediate ill-formed structure.

8 Apart from arguments about learnability. Such arguments are notoriously inconclusive, cf. [Sav87]: "... we are not now in a position to characterize any language class (other than possibly the finite languages) as either provably learnable or provably not learnable."

such a set is one of the research goals in the theory of grammar, and this task can only be solved empirically. Any system will have to choose some such set and will be open to criticism and revision on this count. Furthermore, linguistic theory is incomplete and unsound (in an intuitive sense) with respect to natural language and may even not be consistently formalizable. It is therefore unclear, what *correctness* of a parser with respect to such a theory may mean. The parsing as deduction approach (PAD) suggests one possible answer: In this approach one formalizes grammatical theory in some suitable logic and defines correctness as soundness and completeness (in the sense of formal logic) with respect to this formalization.[9] However, PAD encounters difficulties in explaining ill-formed sentences (which will not be theorems in the logic). So we may prefer to rest with an intuitive definition of correctness based directly on the relation the parser bears to linguistic theory:[10]

- *Correctness:*
  The parser will return the same analyses for grammatical input and the same explanations for ungrammatical input as a competent linguist.

Of course, correctness will in general not be a verifiable property of a PBP system, because program verification in general is impossible. Instead, the coverage and practical usefulness of competing systems will have to be measured by comparing their performance on a corpus of linguistic data. At present, such comparison of competing theories (and parsers) is difficult. First, there is no established test fragment for any natural language, so every implementor will have to invent his own. Second, example sentences in the literature tend to be contrived and complicated, dealing in depth with very special constructions, but disregarding ordinary continued text. A parser geared only to analyze such data will fare badly in everyday life. Here is a point where the interests of the theoretical linguist and of the writer of grammars for NL systems are potentially in conflict with each other. Third, the question of how to organize the material in such a database in a theory-neutral way must be addressed. Often, a system is not intended for wholesale coverage, but without theory-neutral organization of material, partially correct system will be difficult to characterize with respect to coverage.

---

9 Of course, depending on the logic one chooses (e.g. Horn logic, first order predicate logic or whatnot), correctness may become an undecidable property.

10 A definition like this was first proposed by Fong ([Fong91]).

4 Architectures for PBP systems

GB grammars encode linguistic claims about what constitutes knowledge of language, but not about the way humans exploit this knowledge. Of course, as Abney ([Abn85]) observed:

> "A theory of grammar would be of little interest if it were manifestly incapable of supporting reasonable models of linguistic behavior."

The issue of parser control is therefore of great importance for the plausibility of GB as a psychologically realistic theory. It is also more or less independent from considerations of linguistic competence.[11] Let us now review three computational models for PBP that have been documented in recent work.

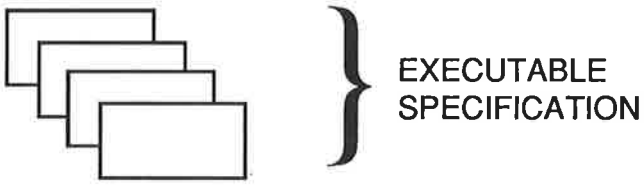

Figure 1: The Executable Specification Model (ESP)

In the ESP model (see figure 1), every module of grammar is specified in logic (typically, Horn logic) and then used immediately as input to a general theorem prover. This idea is due to Mark Johnson ([John88, John89]). Unsurprisingly, it inherits the difficulties associated with the incompleteness of automated theorem provers: Left-recursion or ε-productions may throw it off, negation is typically unsound, and so on. The ESP also has an unfortunate tendency towards generate-and-test algorithms. Johnson proposes to use program transformations like unfold/fold and co-routining by goal freezing[12] to cope with these problems. The power of these methods is limited, however, and the transformed grammar may still have to be recoded by hand to be executable. An alternative description of the problem is that the idiosyncracies of one's theorem prover introduce extraneous restrictions on the formalization of the principles. Finally, there is no place in the ESP to model performance aspects, e.g. preferences in PP-attachment and the like. Historically, Johnson's work has been very influential, and particularly the

---

11 But see section 2.

12 Freezing is an alternative control strategy for Prolog, where the execution of a goal is made conditional on the instantiation of specified variables. It is familiar from Prolog-II and Sicstus.

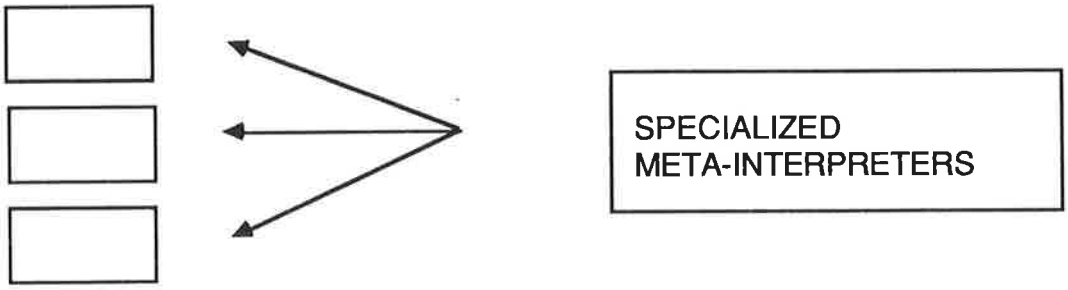

Figure 2: The Multiple Interpreters Model (MIM)

use of a declarative formalism with goal freezing is a hallmark of many PBP systems now around, including those incorporating the MIM and DUM models discussed next.

With the MIM (see figure 2), it is not the principles which are executed and co-routined, but rather special-purpose inference engines which exploit the knowledge formalized in the modules of grammar. The model was first introduced by Crocker ([Cro91]). In the extreme case, there may be one interpreter for each module. This model is more flexible than ESP in at least two respects: First, different proof strategies may be appropriate to different principle systems. Second, performance aspects can be modelled in the proof strategies, e.g. by including selection rules for the application of principles (as proposed in [Fong91]). Selection rules may be dynamic, i. e. sensitive to the history of the proof. The main danger in implementing such a system is that too much linguistic knowledge may actually be coded in the interpreters, by inadvertently or purposefully hand-compiling parts of the grammar away. Changes in the grammar would have non-local repercussions.

The DUM (see figure 3) does not suffer from the last-mentioned defect, because the parsing strategy can access the grammar component of the system only via an interface. The model effectively isolates the grammar from the parser. The parsing strategy is to be thought of as completely grammar-independent. It will use very general structure-building instructions only (like "get the next input

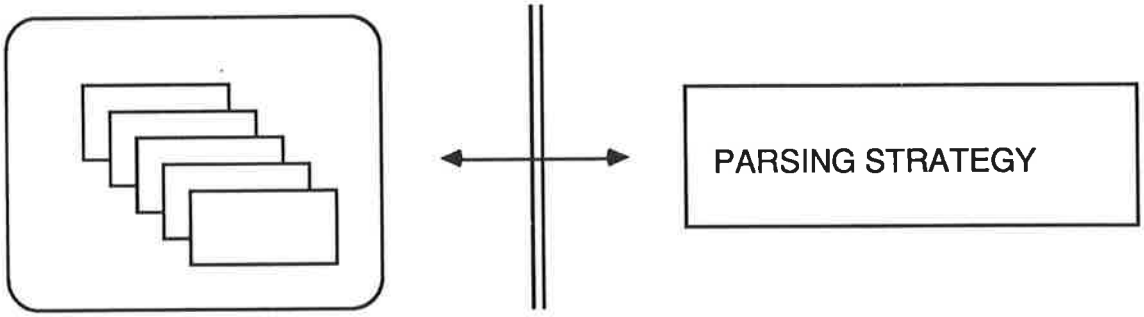

Figure 3: The Dual Model (DUM)

word", "have a projection made") and pass structures to the grammar component to check for well-formedness. In this model, both the principles and the parsing strategy will be interleaved. Such a system can be very flexible, but may also be impure: Only the grammar need be logically formalized, while the parsing module is completely unrestricted and may not even have a declarative semantics. Systems according to this model were introduced by [Mil90] and [Mac91]. To conclude: Both the MIM and the DUM are viable alternatives for practical systems, both with their own advantages and shortcomings.

While we have been concerned with the relation of the representation of linguistic knowledge to the procedural control of the parser, we will now briefly consider the interactions between the principles themselves. Of course, the desire for a modular system will make the implementor strive to minimize such interaction. If every module operated only on its own representation, without regard to other modules, a real parallel implementation of PBP might even be possible. In all current systems known to the author, on the other hand, *all* modules have access to phrase structure (PS),[13] and there is usually at least one special representation in addition to PS, namely for chains. This has proved useful in guiding the insertion of empty categories and enforcing global grammatical constraints ([Cro91], [Fra90], [Mil90]). We may consider the universal accessibility of PS a defect, because each principle should only be concerned with certain features of it. Again, indexing techniques can be used to "condense" certain properties of PS into a new representation which is then passed to the appropriate module ([Mil90]). More thought needs to be given to the creation of well-defined interfaces between modules.

Many systems adopt an interleaving and licensing approach to parsing simultaneously. The licensing approach goes back to [Abn86], who states: "A structure is well-formed only if every element in it is licensed." In an interleaving strategy, licensing conditions will be activated as early as possible to avoid building any ungrammatical structure. This approach is found most often in a parsing as deduction approach, where the emphasis is on efficient derivation of grammatical structure. It is less suited to the intuitive standard of correctness proposed above, which demands explanations of ill-formed input. Similar observations can be made with regard to derived principles, i.e. principles derived from a set of first principles by (hand-)compiling them away, or explicitly asserting some of their consequences. E.g., elementary consequences of case- and θ-theory, such as that

13 It is suggested in [Cro91] that every module construct its own representation only. However, Crocker's implementation as well gives every module access to phrase structure.

case-bearing items without θ–role head an NP-chain, are sometimes used immediately to guide the process of chain formation. Again, the advantage is efficiency. The disadvantage in this case is twofold. First, the capacity for adequately explaining why a structure is ill-formed is lost, because principles derived from several sources are too coarse-grained. Second, maintenance will become difficult, because changes in proper principles will necessitate revision of the entire set of derived principles. This kind of efficiency-mongering should therefore not be part of the design of research tools. Of course, when ultimately an application system is sought after, an existing parser may be compiled out to yield a static, inflexible, and efficient system.

4 Levels of Representation

While GB is distinguishes several levels of representation, GB-parsers are typically *monostratal.* It has often been noted that there is no convincing evidence for D-structure as a psychologically real level of representation. Talk of movement is regarded as metaphorical, with move-α being just one more static relation in the syntactic structure. Johnson has shown how constraints on D-structure can be folded into a deductive parser without actually having to construct D-structure itself. So the greatest omission in those systems seems to be the absence of a level of Logical Form (LF). Interestingly, recent work on the syntax-semantics interface suggests that LF is not needed for many purposes for which it was once thought necessary, in particular with regard to quantifier scope and anaphoric binding. Even more important is the insight, that the sequential model of syntactic-semantic processing inherent in GB's conception of LF is necessarily inadequate. Pinkal convincingly argues in [Pin91] that some syntactic constraints on anaphoric binding can only be enforced after a certain amount of semantic processing has already taken place. At Saarbrücken University, we seek to develop an S-structure based module for semantic interpretation, consisting of very general, declarative principles of semantic interpretation, not making reference to or duplicating parts of the syntactic structure, and a set of semantic objects (denotations) associated with positions in the syntax tree. This module will be co-routined with the syntactic modules, so that the application of interpretation rules can be sensitive to the syntactic environment and vice versa. If this is feasible, it will provide a point of comparison to another well-established paradigm (which one may call principle-based in a wider sense), namely HPSG.

5 Conclusion

Several modular, declarative, faithful, and direct implementations of GB-parsers have been developed in the past few years. Progress has been made in rigorously formalizing GB, investigating the consequences of the parsing-as-deduction paradigm for PBP, discovering advantages and disadvantages of several control strategies with regard to different (and sometimes conflicting) desiderata for parser behaviour, and in developing guidelines for plausible architectures. There exist prototypical, tool-box like system for the development of GB-based grammars. It is hoped that principle-based approaches to parsing will help to elucidate the human language faculty, as well as help to bridge the gap between computational and theoretical linguists.

6 References

[Abn85] Abney, S. & Cole, J. (1985), *A Government-Binding Parser,* North Eastern Linguistic Society 16, pp. 1--17.

[Abn86] Abney, S. (1986) *Licensing and Parsing,* North Eastern Linguistic Society 17, pp. 1--15.

[Chom81] Chomsky, N. (1981), *Lectures on Government and Binding,* Foris, Dordrecht.

[Chom86] Chomsky, N. (1986), *Barriers,* MIT Press, Cambridge, Ma.

[Cro91] Crocker, M. W. (1991), *Multiple Interpreters in in a Principle-Based Model of Sentence Processing,* EACL Proceedings 5, Berlin.

[Fong91] Fong, S. (1991), *Computational Properties of Principle-Based Grammatical Theories,* unpubl. diss., MIT, Cambridge, Ma.

[Fra90] Frank, R. (1990), *Licensing and Tree Adjoining Grammar in Government Binding Parsing,* Ms., GB-Parsing Workshop, Université de Genevé.

[John88] Johnson, M. (1988), *Deductive Parsing with Multiple Levels of Representations,* ACL Proceedings 26, Buffalo, NY.

[John89] Johnson, M. (1989), *Parsing as Deduction: The Use of Knowledge of Language,* Journal of Psycholinguistic Research, Vol. 18, No. 1.

[KT 91] Kolb, H.-P. & Thiersch, C. (1991), *Levels and Empty Categories in a Principles-and-Parameters Approach to Parsing,* in: Haider, H. & Netter, K. (eds.), *Representation and Derivation in the Theory of Grammar,* Kluwer Academic Press, Dordrecht.

[Lat91] Latecki, L. (1991), *An Indexing Technique for Implementing Command Relations,* EACL Proceedings 5, Berlin.

[Mac91] Macias, B. (1991), *An Incremental Parser for Government-Binding Theory,* unpubl. diss., Wolfson College, Cambridge.

[Mil90] Millies, S. (1990), *Ein modularer Ansatz für prinzipienbasiertes Parsing,* IWBS Report 139, IBM Germany Ltd, Stuttgart.

[Pin91] Pinkal, M. (1991), *On the Syntactic-Semantic Analysis of Bound Anaphora*, EACL Proceedings 5, Berlin.

[Sav87] Savitch, W. (1987), *Theories of Language Learnability,* in: Manaster-Ramer (ed.), *Mathematics of Language,* John Benjamins Publishing Company, Amsterdam/Philadelphia.

[Sel85] Sells, P. (1985), *Lectures on Contemporary Syntactic Theories,* CSLI Lecture Notes No. 3, CSLI, Stanford, Ca.

[Sta89] Stabler, E. (1989), *The Logical Approach to Syntax,* MIT Press, Cambridge, Ma. (forthcoming)