MODULARITY, PARALLELISM, AND LICENSING IN A PRINCIPLE-BASED PARSER FOR GERMAN

SEBASTIAN MILLIES
Saarbrücken University
Dept. of Computational Linguistics
D-6600 Saarbrücken
Germany
e-mail: millies@coli.uni-sb.de

Abstract

This paper presents a direct implementation of Government-Binding theory in a parser for German, which faithfully models the modular structure of the theory. The modular design yields a flexible environment, in which it is possible to define and test various versions of principles and parameters. The several modules of linguistic theory and the parser proper are interleaved in parallel fashion for early elimination of ungrammatical structures. Efficient processing of global constraints is made possible by the concept of licensing, and the use of tree indexing techniques.

1 Introduction

Government-Binding theory[1] (henceforth "GB") seeks to describe human knowledge of language by positing a small number of highly general principles, which interact to produce highly specific effects. Most of these principles are regarded as universal principles. Specific construction types in different human languages result from applying language-particular versions of the universal principles, derived from them by parametrization. GB tries to avoid language-particular and construction specific rules. Only recently has the idea of "principle-based" parsers, which derive structures by deduction from an explicit representation of the principles, come into the focus of attention. Importantly, however, GB does not specify any particular relation between the principles and a parser which is supposed to use them. As a consequence, extant GB-parsers reflect the internal organization of GB-theory to varying degrees. This paper reports on an implementation of a GB-parser for German, which faithfully mirrors the modular structure of (much of) GB-theory in the way it represents linguistic knowledge. In discussing the parser, I will presuppose a basic familiarity with GB-theory.[2]

According to Mark Johnson (cf. [John88, John89]), the most direct relation between a parser and linguistic theory can be observed in a "parsing-as-deduction" approach. Johnson's project is to formalize linguistics in some suitable subset of first-order logic, and use this formalization as input for an automatic theorem prover, such as Prolog, without any intervening recoding. This proposal, however, suffers from some well-known difficulties, such as undecidability, left-recursion (in Prolog), and a tendency to produce generate-and-test algo-

[1] By that term I will mean not only the particular version of the theory set forth in [Chom81], but rather the entire family of theories of the principles-and-parameters type inspired by Chomsky's work.

[2] The reader is referred to [Sel85] for a short introduction. For a detailed discussion, see one of the standard texts, e.g. [LU88].

rithms (with modules such as X'-theory and move-α as generators, and other parts of grammar as filters). Furthermore, there is no place in the model for those aspects of language processing which do not have to do with knowledge of grammar, but rather with procedural considerations (resolution of ambiguities in PP-attachment and the like).

Johnson proposes to cope with the difficulty about indeterminacy by using the *freeze* -construct (known, e.g. from Prolog-II) to achieve pseudo-parallel execution of generators and tests. The *freeze* control structure suspends the execution of goals depending on the instantiation of specified variables. This relaxes some of the procedural constraints on the formulation of logic programs, and brings out the logical structure of a program more forcefully. The current approach is similar to Johnson's in that it also uses a formalization of linguistic principles in Horn logic, and executes this formalization in a parallel fashion using *freeze*. It differs from that approach, in that the principles do not themselves constitute the parser, but rather work in tandem with a specialized module, which implements the procedural aspects of parsing. Indeterminacy in the linguistic component is further reduced by having lexical information constrain X'-theory from being fully productive, and using an extension to the concept of "licensing" ([Abn86]) to guide the introduction of empty categories. The total effect is to allow the formalization of the theory to be maximally declarative, and at the same time to ensure decidability of the parsing problem for all possible input. Another key idea is to use clever indexing techniques on trees for the efficient enforcement of conditions on potentially arbitrarily large parts of the parse-tree (e.g., subjacency, or the ECP).

2 Implementation of a GB-Parser

Figure 1 is a (slightly simplified) schema of the system architecture. The entire system has been programmed on an IBM RT in Quintus-Prolog 2.4 under Unix/AIX. As Quintus does not have a *freeze* predicate, a meta-interpreter has been implemented to provide one. The interpreter is fully transparent to the grammar designer; in particular, it handles the cut, and knows about Quintus' module concept. The schema makes the modular organization of the system very clear.

This kind of modularity makes for a great deal of flexibility. The aim of this work is not just to "hardwire" some particular version of GB into a parser, but rather to provide an environment, where different versions of GB-theoretical grammars can be tested and evaluated. In the program, this aim has been approached closely, as the definitions of the principles are not spread out over several components of the grammar, but are textually localized, and procedurally independent from each other and the parsing module. As a consequence, they can be updated or played around with quite easily. The environment also provides tools for facilitating grammar development, such as functions for installing new sets of parameters, a customizable pretty printer, or a small tracing facility. We will now in turn discuss some of the components shown in Figure 1.

2.1 The Parsing Module

The parsing module is independent from the rest of the system, and can be exchanged for a different module, implementing a different parsing strategy. In this way, it is possible to model performance aspects of human sentence processing without having to change the declarative representation of linguistic knowledge as such. The language- and grammar-independence of the parsing module is manifested by its making use of very general structure-building instructions, which do not mention grammatical notions at all, except on a very high level and in an extremely unspecific manner. All the details of the representation of linguistic knowledge are hidden from the parsing module. Typical instructions are:

- read the next input word
- insert a partial tree into the structure that is being built
- have a maximal projection made
- insert an empty category
- check local/global grammatical constraints

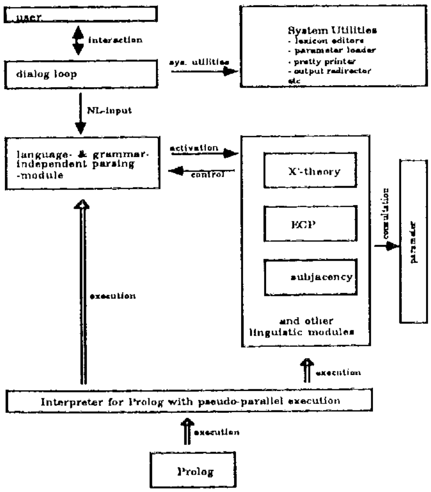

Figure 1

The parser directly reconstructs S-structure. There is no need to view D-structure as a level of representation distinct from S-structure, because D-structural representations are determined on the level of S-structure by the co-indexing of moved constituents with their traces. At present, the parser uses a simple head-driven method of structure building: It proceeds from left to right through the input string, projects every word to the phrasal level, and pushes all projections into a queue until it finds the head of the substructure that is being analyzed. It then inserts this substructure into the analysis tree and tries to empty the queue. E.g., while parsing the sentence *daß Hans Maria liebt* (literally, "that John Mary loves"), the parser will first project *daß* to CP, push two DPs onto the queue, project *liebt* to IP, and then empty the queue. The parser can handle head-complement structures of German. It cannot handle adjunction, which is a serious restriction, to be lifted in later versions of the parser. The types of phenomena currently covered are: Main and subordinate clauses (both V2 and verb-final) nested to arbitrary depth, wh-movement (both direct and indirect questions), infinitives (ECM, Raising, Control), passive, prenominal genitives and adjectival modification, and agreement between determiners, adjectives, nouns, and verbs.

2.2 Linguistic Knowledge

The following modules of GB-theory have been implemented: X'-theory, move-α, case theory, θ-theory, the projection principle, bounding theory, government theory (specifically, a notion of "barrier" (cf. [Chom86]) is included in the definition of the ECP), spec-head-agreement, and spec-head-licensing. X'-theory is constrained to project only nodes licensed by lexical properties of the head (specifically, subcategorization and θ-marking license the projection of argument nodes in a structure).[3] Linguistic constraints are classified according to their potential domain of application into *local* constraints (which apply internal to a phrase) and *global* constraints (which have a potentially unlimited domain of application). Currently, the ECP and the subjacency principle are implemented as examples of global constraints. As for local constraints, there are the Head Feature Principle (similar to GPSG's Head Feature Convention), case-marking, the first half of the θ-criterion (guaranteeing that every argument gets at least one θ-role), L-marking, and spec-head-agreement/licensing. All local constraints are enforced immediately after lexical projection has taken place. This is true also for spec-head-licensing relations: These conditions can be locally activated even before anything is known about the actual content of the specifier position. They will be explicitly consulted only once: Using the *freeze* mechanism, they will afterwards be active in the background, parallel fashion, and will prevent the parser from building any unlicensed structure.

Parameters

The following parameters can be set: The positions of heads and specifiers relative to the complements, the number and categorial identity of bounding nodes (for subjacency), the number and categorial identity of potential barriers, the categorial identity of L-marking heads and lexical heads, and the possibility of V-to-I (I-to-C) movement.[4]

Chain formation and enforcement of global constraints

Case is assigned to chains, so that every chain gets exactly one case. Similarly, every chain is assigned exactly one θ-role. These requirements are known as the "case filter" and the "θ-criterion" resp. - Chains, however, can be arbitrarily long, so that these requirements cannot be locally enforced. The same is true of the subjacency principle and the ECP, which constrain the relation between traces and their antecedents. So there are three different questions to answer:

1. Under what circumstances may traces be introduced?

2. How are chains formed? How are the case filter and θ-criterion enforced on chains?

3. How are subjacency and ECP enforced?

As a first step towards answering these questions, let us accept the following condition (taken from [Abn86]): A structure is well-formed only if every element in it is licensed. Abney takes licensing relations to be unique (i.e., every element is licensed by a unique relation), lexical, and local (i.e., valid under sisterhood). As we observed, the locality requirement obviously will not do. We will relax it by positing principle (L):

(L) Every element in a structure is licensed either locally (in Abney's

---

3 This is not as ad hoc a solution as it may seem. In linguistic literature, it has been suggested several times that phrase-structure is in some way derivative from other notions, such as case- or θ-marking. There is no good reason for viewing X'-theory as an unconstrained generator.

4 This is just stipulated by means of a "parameter". There is no explanation of head-movement in the parser.

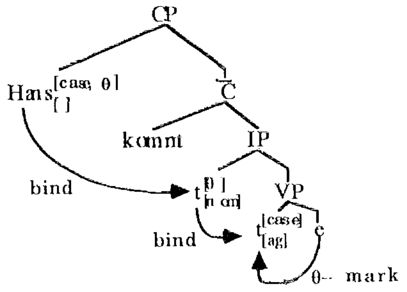

Figure 2

sense), or by locally binding an element which in turn is licensed according to principle (L).

This gives us a way to answer questions 1. and 2.: Arguments and their traces may be introduced into a structure as long as there is a chance that they will end up as local antecedents of some independently licensed trace. Take the case of θ-assignment: In Figure 2, the trace in SpecIP is licensed by virtue of being a local binder of a trace which is licensed by θ-marking, and *Hans* is licensed by binding the trace in SpecIP. This is implemented by putting "requests" for θ-roles in a set associated with each element (requests are noted as superscripts in Figure 2). A θ-request in a chain is satisfied by an element that is θ-marked. The first half of the θ-criterion, which requires every chain to have at least one θ-role, is thus automatically enforced, by positing:

(S) Every request must be satisfied.

The second half of the criterion can be enforced by our putting "offers" for θ-roles on a list as well (subscripts in Figure 2). The offers associated with a chain are determined by multiset union over the offers associated with the chain elements. We then posit that there may be at most one offer per chain. Now, what about case-marking? Obviously, the case filter is so similar to the θ-criterion as to be amenable to the same treatment. However, note that treating case-assignment as a licensing relation in this way is tantamount to giving up Abney's uniqueness condition as well. In Figure 2, *Hans* will be licensed by two relations. A linguist might even want to posit still other licensing relations. So let us put forward the condition of "relative uniqueness":

(RU) Every licensing relation must be offered in a chain at most once.

Taken together, (L), (S), and (RU) answer questions 1. and 2. from above.[5] The solution has been implemented. The actual implementation, however, does not follow the inefficient strategy of constructing chains after waiting for locally licensed traces to appear, but rather reverses the process: The parsing module follows a first-fit strategy, inserting elements top-down in the highest possible position, hypothesizing that these elements will be licensed according to principle (L). These hypotheses (i.e., the presence of unsatisfied requests) license the further appearance of traces in a chain. This method even eliminates the need for explicit chain construction. Instead, requests are simply inherited from the local antecedent down the tree until they are cancelled.[6]

Let us turn to question 3. In doing so, let us also consider how expensive it is to check for sub-

5 R. Frank ([Fra90]) has independently arrived at a similar solution within the framework of TAGs.

6 The module for chain construction can be seen as an interpreter *exploiting* the principles of grammar, which are in this case not used directly in parsing, cf. M. Crocker's discussion of this point in [Cro91].

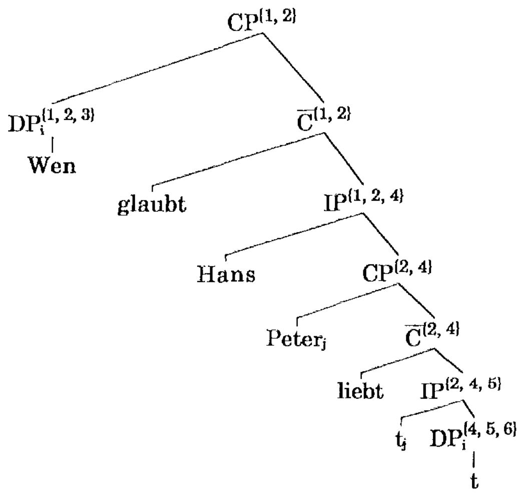

Figure 3

jacency and antecedent government. We shall see that with an indexing scheme on trees the check can be done in log(n) time, where n is the size of the tree.[7] Let us take subjacency as an example. The idea is to label the root of the tree with a set of k+1 indices, where k is the maximal number of bounding nodes that may be crossed by move-α. Indices are inherited down the tree, such that at every bounding node a new, unique index is added, and the oldest index is not passed downwards. Figure 3 illustrates this. The following is then true:

(Subjacency) α is subjacent to β iff the index sets on α and γ are not disjoint, where γ is the lowest common ancestor of α and β.

Nodes in the tree have identifiers that specify a path from the root to the node (as there are only binary trees, these paths are given by sequences of 1's and 0's). Thus, finding the lowest common ancestor of two nodes is no harder than selecting the higher of the nodes. Since the cardinality of the index sets is bounded by k+2, the set comparison can be done in constant time. A similar test has been used to implement antecedent government. The *freeze* -mechanism allows us to uniformly state the instruction for constructing the correct index sets on every node right after that node has been projected, although the actual property of being a barrier can only be established after the node has found its definitive place in the parse-tree. Antecedent government can be tested even before all global properties of the tree are known. The following piece of code implements antecedent government (apart from co-indexing). It demonstrates the elegance of our modular approach:

```
antecedent_govern(Node1, Node2) :-
     node_info(IndB1, Node1),
     node_info(IndB2, Node2),
     freeze(IndB1,freeze(IndB2,
          \+disjoint(IndB1,IndB2))).
```

[7] Indexing schemes were originally developed by L. Latecki for the analysis of scope ambiguities and command relations ([Lat91]).

3 Conclusion

A modular implementation of a government-binding parser for a considerable fragment of German has been outlined. A new concept of licensing, the use of indexing techniques, and the pseudo-parallel interleaving of a parsing strategy with a faithful, direct, and declarative representation of GB-theory have led to a prototypical, tool-box like system for the development of GB-based grammars. The system has been fully implemented in Quintus-Prolog. It is hoped that principle-based approaches to parsing will help to elucidate the human language faculty, as well as provide a novel focus for the approaches of both theoretical and computational linguists.

4 References

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

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

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

[Coh85] Cohen, J. (1985), *Describing Prolog by its Interpretation and Compilation*, Communications of the ACM, Vol. 28, No. 12.

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

[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.

[LU88], Lasnik, H., Uriagereka, J. (1988), *A Course in GB Syntax*, MIT Press, Cambridge, Ma.

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

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

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