Parsing with ID/LP and PS rules

Volk, Martin

Parsing with ID/LP- and PS-rules

Published in “Proceedings of the 3rd KONVENS Conference” (Bielefeld, 1996)

Martin Volk
University of Zurich,
Department of Computer Science, Computational Linguistics
Winterthurerstr. 190, CH-8057 Zurich
volk@ifi.unizh.ch

Abstract

ID/LP-grammars have been proposed as elegant formalisms for natural language syntax. They are said to be superior to phrase structure (PS) rules in that they allow the explicit distinction between dominance and precedence relations. But this distinction entails difficulties in grammar writing and in efficient processing. We therefore propose a hybrid approach combining ID/LP- and PS-rules in a grammar. We show that parsing with such a hybrid grammar is more efficient than parsing with the corresponding ID/LP-grammar.

ID/LP-Grammatiken wurden als elegante Formalismen zur Beschreibung der Syntax natürlicher Sprachen eingeführt. Es wird behauptet, dass sie Phrasenstrukturregeln (PS-Regeln) überlegen sind, da sie eine explizite Aufteilung in Dominanz- und Präzedenzbeziehungen vorsehen. Aber gerade diese Aufteilung führt zu Schwierigkeiten beim Schreiben von Grammatiken und bei der effizienten Verarbeitung. Wir schlagen deshalb einen hybriden Ansatz vor, der ID/LP- und PS-Regeln integriert. Wir zeigen, dass eine solche hybride Grammatik zu effizienterer Verarbeitung führt als die entsprechende ID/LP-Grammatik.

1 Introduction

Starting with the work on GPSG [GKPS85] it has frequently been suggested that immediate dominance (ID) and linear precedence (LP) rules are superior to regular phrase structure (PS) rules for natural language parsing. This trend continued in work on HPSG [PS94]. The reasons put forward centered around the additional level of abstraction introduced by the distinction between ID- and LP-rules. Traditional phrase structure rules that combine dominance and precedence have been largely disregarded. This move has had the result that even constructs that can be described much easier with PS-rules have to be described with ID- and LP-rules. I will therefore argue that a combination of ID/LP- and PS-rules is advantageous for grammar engineering and does even allow more efficient parsing than pure ID/LP-parsing.

We do not consider ID/LP-parsing within any particular grammatical theory. But we assume that the constituents of the grammar rules are complex feature structures that can be modified via unification. For explanatory reasons we use atomic symbols in our examples, but complications that derive from feature structures will be discussed.

Two arguments have been most prominent in advocating ID/LP-grammars:

1. The distinction between dominance and precedence allows to explicitly state linguistic generalizations (e.g. that a head in English always precedes its complements).

2. ID-rules facilitate the description of variable word order languages (e.g. the middlefield (MF) of a German sentence can consist of nominative-NP, accusative-NP and dative-NP in any order. Assuming a flat sentence structure, one ID-rule suffices to describe the constituents in any order. The same would have required 6 PS-rules).

```
MF --> Aux(finite), NP(nom), NP(dat), NP(acc),  Verb(infinite)
       wird         Peter    ihm      den Ball  geben
       will         Peter    him      the ball  give
       Will Peter give him the ball?
```

Following this argumentation it has been proposed to use ID/LP-grammars instead of PS-grammars. But the proponents have overlooked the computational and engineering problems that come with ID/LP-grammars.

1. ID/LP-parsers are not as efficient as PS-parsers (cp. [BBR87]). The reasons for the additional overhead lie in the fact that the unspecified order in ID-rules makes precise rule invocation (e.g. the prediction of the next symbol) more complicated. In addition linear precedence has to be checked separately.

2. Some ordering phenomena cannot be adequately described by ID/LP-rules. E.g. it is not possible to have a strongly equivalent ID/LP-grammar describing the PS-rule

   ```
   NP --> N1 Conj N1
   ```

   It is not possible to state in ID/LP-terms that the conjunction is located between the two N1s (i.e. to disallow the conjunction to stand at the beginning or at the end of the sequence). Even if the sequence appears in the intended order, the ID-rule has the unwanted side effect that it stands for duplicate readings. This follows from the fact that the parser cannot distinguish between the two N1s and interprets them in either order.[1]

3. It is very cumbersome to write LP-rules for all ID-rules describing linguistic units with no variation in precedence. E.g., while there is considerable variation in constituent order in a German sentence, there is hardly any order variation in a German noun phrase. I.e. all ID-rules describing German noun phrases need to be carefully complemented with corresponding LP-rules.[2]

   ```
   ID 1:  NP --> Det, N
   ID 2:  NP --> Det, Adj, N
   ID 3:  NP --> Det, N, NP(genitive)

   LP 1:  Det < N
   LP 2:  Det < Adj
   LP 3:  Adj < N
   LP 4:  N < NP(genitive)
   ```

[1] The parser can, of course, check for duplicate chart entries and suppress these.

[2] The first LP-rule in this example is redundant. Because LP-rules are transitive by definition, LP-rule 1 could be inferred from the LP-rules 2 and 3. The transitivity of LP-rules is another frequent source of errors in writing ID/LP-grammars. Its effect on a set of some dozen LP-rules are hard to predict by a human. It requires special tools to compute the transitive closure and to check for prohibitive loops (such as $A < B, B < C, C < A$).

I therefore propose to use a hybrid ID/LP- and PS-grammar for all practical purposes. While this approach looses some linguistic generalizations it solves many engineering problems:

- Such a hybrid grammar allows to state all variations with ID/LP-rules and all fixed order phenomena with PS-rules.
- PS-rules can be used for symmetrical sequences that cannot be captured with ID/LP-rules such as

  ```
  NP --> N1 Conj N1
  ```

  avoiding the problem of order specification and of duplicate structures.
- Grammar writing becomes easier since there is no need to specify LP-rules for fixed order phrases such as German noun phrases.

A combination of ID/LP- and PS-rules provides the flexibility needed in an engineering environment in that it provides adequate means for both fixed order and variable order phenomena.

2 Efficiency of a hybrid ID/LP- and PS-parser

It is to be shown now that parsing with a hybrid ID/LP- and PS-grammar can be at least as efficient as parsing with an ID/LP-grammar. Many parsing algorithms for ID/LP-grammars have been proposed. Most are based on Earley's top-down algorithm for context-free grammars which was first modified by Shieber [Shi84] to handle ID/LP-grammars and lateron by Seiffert [Sei87] to handle unification-based ID/LP-grammars. In contrast, Kilbury [Kil84] introduced a bottom-up algorithm for ID/LP-grammars which was modified by Weisweber [Wei87]. I will here concentrate on Weisweber's bottom-up chart parser[3]. This algorithm is efficient for most natural language phenomena which is achieved by using a so called dominance set (derived from the immediate dominance rules). Even if parsing the whole sentence fails, the chart built up by Weisweber's algorithm contains all possible partial solutions.

In order to demonstrate that parsing with a hybrid ID/LP- and PS-grammar does have advantages over pure ID/LP-parsing I will show how this ID/LP-parser can be turned into a hybrid ID/LP-PS-parser. Weisweber himself had noted that his algorithm could easily be transformed into a PS-parser ([Wei87] p. 45), but he did not investigate the advantages of a hybrid parser. My argumentation for ID/LP-PS-parsing does not rely on the parser being bottom-up and should therefore be applicable to other parsers as well.

Weisweber's parser uses dominance sets to speed up parsing. A dominance set is collected from the dominance relation, which in turn is made explicit by asserting every single dominance relation before parsing starts. That means, that for two arbitrarily numbered ID-rules such as:

```
1:  A --> B, C, D
2:  E --> F, C
```

the following dominance relations will be saved:

```
dom(B, 1).  dom(C, 1).  dom(D, 1).
dom(F, 2).  dom(C, 2).
```

[3]To be more precise: It is a well-formed substring table parser since the chart contains only complete edges.

This says that the constituents B, C, and D are dominated in rule 1 and that F and C are dominated in rule 2. From these relations the following dominance sets will be gathered:

```
DomSet(B) = {1}  DomSet(C) = {1,2}
DomSet(D) = {1}  DomSet(F)= {2}
```

There is one dominance set for every constituent in the grammar, except for the start symbol, which is not dominated by any other constituent. The set for constituent C contains pointers to all the ID-rules where C is dominated. Since the ID-rules are uniquely numbered the dominance set for C contains a set of numbers that stand for specific ID-rules where C is among the constituents of the right hand side (RHS). The dominance set is used in the reduction step of the parser. The reduction step applies a rule from right to left. When all elements of the RHS are found, this sequence can be reduced to the mother symbol.

While in bottom-up PS-parsers a grammar rule will be considered for reduction when either the first element of its RHS is found (in active chart parsers) or the last element of its RHS is found (in well-formed substring table parsers), the information about the first or last element is not available in ID-rules. By definition an ID-rule's RHS is unordered. Therefore it is necessary to consider a rule for reduction when any element of its RHS is found. But searching through every rule at every reduction step is too costly. This is where the dominance set comes in to quickly find the relevant rules. Thus the union of all dominance sets can be seen as an index over the RHSs of the ID-rules.

If the grammar contained only atomic symbols we would compute the dominance sets for all grammar symbols before parsing starts. But since we are working in a unification based formalism, the grammar symbols contain complex feature structures. During parsing a feature structure can be modified by unifying it with other feature structures. A feature structure can thus be enriched with information that can influence membership in the dominance set. Therefore, collecting the dominance set needs to be delayed as long as possible and is performed at run-time immediately before the set is needed by the parser.

So here is how Weisweber's algorithm works with the dominance relation:

```
For every word W in the input
where W is located between the positions From and To do
  for every category Cat of W do
    process(Cat, From, To)

process(Cat, From, To)
  get DomSet(Cat)
  save chart(Cat, From, To, DomSet)
  reduce(Cat, From, To, DomSet)

reduce(Cat, From, To, DomSet)
  for every ID-rule in DomSet do
    try to reduce Cat by
     finding the other Categories of the RHS in the chart
     checking LP-rules on the complete RHS
  if reduction to Mother is successful then
    process(Mother, NewFrom, To)
```

When this process is finished the chart contains all complete edges. If there is an edge spanning the complete input, the parse was successful. Weisweber claims that this algorithm is of complexity $O(n^2)$ for grammars without left and right recursion. If the grammar contains right recursive rules the complexity is $O(n^3)$ and it is exponential if the grammar contains left and right recursion.

It can easily be shown that the algorithm is wasting effort in trying to reduce every category with every ID-rule where it is dominated. Let us assume that parsing has begun with finding the category `B`. Then the parser will attempt to apply `B` in a reduction step using e.g. the rule

```
A --> B, C, D
```

but `C` and `D` have not been found and therefore the reduction will fail. To make this a real example, consider the case where the parser has found a determiner. It will then immediately try to apply the rule

```
NP --> Det, Adj, N
```

although `Adj` and `N` have not been found yet. Reduction will fail again when the `Adj` has been found and the `N` is still missing. In sum, the parser disregards the fact that a rule can only be used for reduction when all its constituents on the RHS have been found. While it is difficult to employ this knowledge with ID-rules (though it is partially possible as we will see under section 3.1), it is easy to use this knowledge with PS-rules. A PS-rule can only be used for reduction in such a parser when the last constituent on its RHS has been found. Therefore only this constituent will be entered into the dominance set preventing the example rule to be tried with only `Det` or `Det-Adj`.

In this way a PS-rule with n constituents on its RHS reduces the reduction attempts from n to 1. In addition, it does not need LP checking. As soon as a PS-rule's RHS is found, reduction can take place.

2.1 Modification of the ID/LP-parser

In order to profit from the advantages of PS-rules we have to modify the parser such that

1. it knows how to react on PS-rules and that
2. it can easily switch between applying ID-rules and PS-rules.

It is crucial for the correct working of the hybrid ID/LP-PS-parser that the dominance set is established with information about which rule is an ID-rule and which rule is a PS-rule. Let's assume, that we have an ID-rule and a PS-rule such as:

```
ID 1:  A --> B, C, D
PS 2:  E --> F C
```

the following dominance relations will be saved:

```
dom(B, id(1)).  dom(C, id(1)).  dom(D, id(1)).
dom(C, ps(2)).
```

This says that `B, C` and `D` are dominated in ID-rule 1 and that `C` is dominated in PS-rule 2. Please note that there is no entry for `F` since the PS-rule can only be applied after `C` has been found. Furthermore, the type of the rule has been added to the rule number in the dominance relation. Resulting from these relations the dominance set for `C` would look like

```
 DomSet(C) = {id(1), ps(2)}
```

So here is how my hybrid ID/LP-PS-algorithm works (modifications to the original are marked):

```
For every word W in the input
where W is located between the positions From and To do
  for every category Cat of W do
    process(Cat, From, To)

process(Cat, From, To)
  get DomSet(Cat)
  save chart(Cat, From, To, DomSet)
  reduce(Cat, From, To, DomSet)

reduce(Cat, From, To, DomSet)
  for every rule in DomSet do
>>  if rule is an ID-rule then
    try to reduce Cat by
      finding the other Categories of the RHS in the chart
      checking LP-rules on the complete RHS
>>  else rule is a PS-rule
>>  try to reduce Cat by
>>    finding the other Categories of the RHS in the chart
>>    in the required order
    if the reduction to Mother is successful do
      process(Mother, NewFrom, To)
```

It can be seen that the first overhead induced by the hybrid approach concerns the compilation of the dominance relation. But this is being done before parsing starts and can thus be neglected. The second overhead concerns the decision on what path to follow in the 'reduce' procedure. But this decision is an easy and deterministic yes/no decision that can be based on the information in the dominance set, since every rule number is annotated with its rule type. So again this additional effort can be neglected. This proves that the hybrid ID/LP-PS parser is at least as good as the underlying ID/LP-parser. But since working with PS-rules is more efficient than working with ID/LP-rules we expect the hyprid parser to have an even superior performance. Let us investigate how big an improvement we can expect.

Following Weisweber we investigate the efficiency of the parser by looking at its components 'process' and 'reduce'. The 'process' procedure is determined by the number of edges that will be added into the chart. Weisweber ([Wei87] p. 46) argues convincingly that for n input symbols there will be edges proportional to n^2. The argument is based on the observation that for position 1 in the input string there can be no more than $c * k$ edges, where k is the number of grammar symbols and c stands for the degree of ambiguity. Any other position p can have $p * c * k$ edges. This adds up to $n * c * k(n + 1)/2$ which is proportional to n^2. Since the hybrid algorithm does not change 'process' we do not have to ponder over this.

For every edge in the chart 'process' calls 'reduce' once. 'reduce' calls itself and tries to find edges to the left of the current symbol. The recursion is bound by the length of the RHS of the rules and independent of the length of the input. Weisweber concludes that therefore 'reduce' contributes to the overall complexity only with a constant r which is the maximum number of constituents on the RHS of any rule. This figure is provided for grammars without any right-recursive rules. If the grammar contains right-recursive rules 'reduce' has the upper bound n (resulting in an overall complexity of $O(n^3)$) and if the grammar has right- and left-recursive rules the overall complexity is exponential.

But Weisweber totally disregards grammar size as a factor of complexity. 'reduce' is not only called recursively for one rule but for all rules in the dominance set. If G is the grammar size (the number of ID-rules in an ID/LP-grammar) G needs to be multiplied with whatever complexity one assumes for 'reduce'.

It is this factor where the move towards a hybrid ID/LP-PS-grammar shows advantages, admittedly not by a whole order of magnitude but by a considerable constant. Let us assume that ID-rules in a natural language grammar have on average 3 elements on their RHSs. So any constituent will in 66 % of the cases be on the first or second position of the RHS. That is, in two out of three cases 'reduce' is called in vain. This does not even consider that a complete RHS can still be ruled out by the LP-rules. So if we could substitute half the ID-rules for an equal number of PS-rules we can expect a 33% increase in successful calls to 'reduce'.

Obviously the gain is higher if we have grammars with a higher average of elements on the RHS. If, on the other hand, we assume the average to be lower than 2, it might be best to multiply out the ID/LP-rules into a PS-grammar. Explosion of grammar size, the main argument brought forward by the proponents of direct parsing with ID/LP-rules, is then no longer a problem.[4]

3 Further improvements

3.1 Using LP-rules in establishing the dominance relation

We have shown that the knowledge about when to employ a rule for reduction can be used to restrict the dominance set for a given constituent. In PS-rules we know the rightmost element on the RHS and can therefore tailor the dominance set accordingly. Continuing this line of thought we show that LP-rules can be used to restrict the invocation of ID-rules.[5] We will first demonstrate this with an example. Let's assume that we have the following grammar with one ID- and one LP-rule.

```
ID 1:  A --> B, C, D
LP: D < B
```

ID-rule 1 stands for six sequences:

```
Seq1:  B, C, D          Seq4:  D, B, C
Seq2:  C, B, D          Seq5:  C, D, B
Seq3:  B, D, C          Seq6:  D, C, B
```

But the sequences `Seq1, Seq2` and `Seq3` are ruled out by the LP-rule. In particular `Seq1` and `Seq2` are of interest for our purpose. These are all and exactly the sequences where `D` is the

[4]It may be worth noting that in search for an efficient implementation of GPSG, Fisher ([Fis89]) has totally dispensed with the ID/LP distinction.

[5]This is similar to Kilbury's [Kil84] algorithm, where Kilbury uses the LP-rules to precompute a FIRST-relation that contains every possible left-most constituent of a RHS. The FIRST-relation is used to improve the predictor.

rightmost symbol. None of the permitted sequences has D as a rightmost symbol. Therefore D can be omitted in the dominance relation of rule 1 since it can never be the rightmost symbol in this rule. The resulting dominance relation will then only consist of:

```
dom(B, id(1)).
dom(C, id(1)).
```

In order to generalize this, we can state the following rule which is to be applied when collecting the dominance relations.

Rule for omission from dominance relation: Given an ID-rule with number N and an LP-rule (with $1 =< i, j =< n$ and $i \neq j$)

$N : A \rightarrow C_1, ..., C_n$
$C_i < C_j$

omit C_i from the dominance relation based on N.

This works fine as long as we have a pure context-free grammar. But if we are using a unification based grammar, things are slightly more complicated since categories can be enriched with information during parsing. The feature structures in LP-rules are static but the feature structures in ID-rules are typically being modified during parsing. If the modification of the feature structures is only performed by unification, we know that information is never subtracted. We can therefore use the above rule with the restriction that the feature structure that comes with C_i and C_j in the ID-rule is more specific than the feature structure that comes with the corresponding constituents in the LP-rule. In other words, a constituent in the LP-rule subsumes its counterpart in the ID-rule. In this way we can make sure that the appropriate LP constraint does apply to the constituent at parse time.

[Sei87] showed that unification can even result in non-local feature passing which makes parse-time LP checking insufficient and requires another LP check after the complete parse tree has been built. But it seems to us that there are only very rare cases where a natural language grammar shows such a behavior. The one example given by [Mor95] (p. 15-17) is linguistically dubious and poses a problem only for right-to-left processing.

3.2 Excluding constituents from the dominance relation

If we treat all fixed order phenomena with PS-rules and all LP-rules are evaluated when compiling the dominance relation for the ID-rules there remain only few cases where the dominance relation contains irrelevant information. One such case is an ID-rule such as:

```
ID 1:  A --> B, C, D
```

where we want to allow only the sequences

```
Seq1:  B, C, D
Seq2:  D, C, B
```

As in the symmetrical sequence example, there is no way to describe this with ID- and LP-rules without introducing an additional symbol. Of course we could resort to using two PS-rules instead of ID/LP-rules. But this would undermine the goal that only fixed order phenomena should be described with PS-rules.

This problem could be tackled with a 'fix' predicate which fixes a constituent on a specific position, thus restricting ID-rules into rules with partially free order. We could then write

```
    ID 1':  A --> B, fix(C), D
```

This rule fixes `C` on the second position while all other constituents of the RHS can vary freely. Such a predicate requires modifications in computing the dominance relation and modifications in the parsing algorithm. A fixed constituent is then entered into the dominance relation only if it is fixed to the rightmost position. In this special case all other constituents of the rule can be omitted from the dominance relation. In all other cases the non-fixed constituents are treated as in a regular ID-rule. The parser, on the other hand, needs to be modified that it ensures the fixed constituent to be at the correct position. This can be checked either in the 'reduce' procedure or during the application of the LP-constraints.

Continuing this line of thought one can dream up predicates that restrict a constituent to a certain region in the sequence such as 'not_last', 'not_first'. But I believe that these predicates will obscure grammar writing and will result in an inacceptable overhead in the parsing algorithm.

Purists may also argue that the introduction of such predicates into ID-rules destroys the clear separation of dominance and precedence in ID/LP-grammars. But our move towards hybrid ID/LP-PS-grammars takes into account a weakening of this distinction for the benefit of more adequate and concise grammatical descriptions and more efficient parsing.

4 Conclusion

I have argued that grammar writing in an engineering environment will profit from a combination of ID/LP- and PS-rules. It makes grammar writing easier and generally results in an increased processing efficiency. The improved efficiency has been demonstrated for Weisweber's ID/LP algorithm but I believe that the arguments are equally applicable to other ID/LP algorithms, because all that is required is a deterministic decision of having the parser switch between the treatment of ID- and PS-rules. Our results are in line with findings by [Wir88] who compared different active edge chart parsers: "The experiments clearly indicate that it is possible to significantly increase the efficiency in chart parsing by fine-tuning the rule-invocation strategy." Since PS-rules allow for an improved rule-invocation over ID/LP-grammars these findings support the move towards hybrid ID/LP-PS-grammars.

The hybrid parser has been implemented in SICStus Prolog on a SUN workstation. A test grammar has been set up with 35 ID-rules and 8 LP-rules describing a flat sentence structure for intransitive, transitive, and ditransitive verbs. 12 of these ID-rules are dealing with NP-syntax, 10 are describing the general sentence structure, the others are for verb groups (6), adjective phrases (4), and prepositional phrases (3). The grammar was then translated into a strongly equivalent hybrid ID/LP-PS-grammar resulting in 10 ID-rules, 2 LP-rules, and 25 PS-rules. A sentence with 3 simple NPs such as

```
Hat der Vater seinem Sohn einen Ball geschenkt?
has the father his   son   a    ball  given-as-present
Has the father given a ball to his son as a present?
```

is being processed 25% faster with the hybrid parser (measured in CPU seconds). If the NPs are more complex, containing adjective and genitive attributes, the speed-up is even higher (in the 30% range for 3 complex NPs). These figures must be taken with great care. Speed-up increases when the percentage of ID-rules grows that can be translated one-to-one into PS-rules.

It remains to be shown that a hybrid ID/LP-PS-parser can maintain its advantages when extended to handle optional categories or probabilities in rule application. First practical results show that the treatment of optional categories is not significantly influenced by the hybrid approach. We have still to try the treatment of probabilities. They add another dimension of problems as their application might be dependent on the constituent ordering.

A further line of research should investigate how the algorithm can be used with top-down prediction and with active edges both of which have proven advantageous for PS-grammars. If the reachability relation needed for top-down prediction is computed before parsing, it should be applicable to ID-rules in the same manner as to PS-rules. Using active edges is more costly with ID-rules than with PS-rules as can be seen comparing Shieber's to Earley's algorithm. Hybrid parsing will therefore have an advantage in such an algorithm.

Acknowledgement: The author acknowledges comments and helpful suggestions by Dirk Richarz and Rolf Schwitter on earlier versions of this paper.

References

[BBR87] G.E. Barton, R.C. Berwick, and E.S. Ristad. *Computational Complexity and Natural Language.* The MIT Press, Cambridge,MA, 1987.

[Fis89] Anthony J. Fisher. Practical Parsing of Generalized Phrase Structure Grammars. *Computational Linguistics*, 15(3), 1989.

[GKPS85] Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized phrase structure grammar.* Harvard University Press, Cambridge,MA, 1985.

[Kil84] J. Kilbury. Earley-basierte Algorithmen für direktes Parsen mit ID/LP-Grammatiken. KIT Report 16, TU Berlin, 1984.

[Mor95] Frank Morawietz. Formalization and parsing of typed unification-based ID/LP grammars. Arbeitspapiere des Sonderforschungsbereichs 340, Universität Tübingen, 1995.

[PS94] Carl Pollard and Ivan Sag. *Head-Driven Phrase Structure Grammar.* University of Chicago Press, Chicago, 1994.

[Sei87] Roland Seiffert. Chart-parsing of unification-based grammars with ID/LP-rules. LILOG-Report 22, IBM Deutschland GmbH, Stuttgart, 1987.

[Shi84] Stuart M. Shieber. Direct parsing of ID/LP grammars. *Linguistics and Philosophy*, (7):135–154, 1984.

[Wei87] Wilhelm Weisweber. Ein Dominanz-Chart-Parser für generalisierte Phrasenstrukturgrammatiken. KIT Report 45, TU Berlin, 1987.

[Wir88] Mats Wirén. On control strategies and incrementality in unification-based chart parsing. Thesis no 140, Department of Computer and Information Science, Linköping University, Linköping, Schweden, 1988.