CONTROLLED ACTIVE PROCEDURES AS A TOOL FOR LINGUISTIC ENGINEERING

Heinz-Dirk Luckhardt
Manfred Thiel
Sonderforschungsbereich 100
"Elektronische Sprachforschung"
Universität des Saarlandes
D-6600 Saarbrücken 11
Bundesrepublik Deutschland

Abstract

Controlled active procedures are productions that are grouped under and activated by units called 'scouts'. Scouts are controlled by units called 'missions', which also select relevant sections from the data structure for rule application. Following the problem reduction method, the parsing problem is subdivided into ever smaller subproblems, each one of which is represented by a mission. The elementary problems are represented by scouts. The CAP grammar formalism is based on experience gained with natural language (NL) analysis and translation by computer in the Sonderforschungsbereich 100 at the University of Saarbrücken over the past twelve years and dictated by the wish to develop an efficient parser for random NL texts on a sound theoretical basis. The idea has ripened in discussions with colleagues from the EUROTRA-project and is based on what Heinz-Dieter Maas has developed in the framework of the SUSY-II system.

In the present paper, CAP is introduced as a means of linguistic engineering (cf. Simmons 1985), which covers aspects like rule writing, parsing strategies, syntactic and semantic representation of meaning, representation of lexical knowledge etc.

Survey of some ideas behind CAP

The data structure used in CAP is a type of chart called S-graph (see Maas 1985). Charts are used in parsing quite frequently (cf. Kay 1977, Varile 1983). The S-graph is an acyclic directed graph with exactly one start node and one end node. Each arc carries non-structural information and may carry structural information that is also represented as an S-graph. The non-structural information is a set of property/value-pairs called 'decoration'. It includes

a) a <u>morphosyntactic type</u> (MS), i.e. the terminal or non-terminal category
b) a <u>surface-syntactic function</u> (SF)
c) a <u>deep-syntactic function</u> (DSF)
d) a <u>semantic relation</u> (SR)
e) a <u>weight</u>
f) information specific to an MS

The structure of the complex NP 'trouble with Max' is visible to the user as Fig. 1.

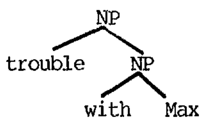


Fig. 1

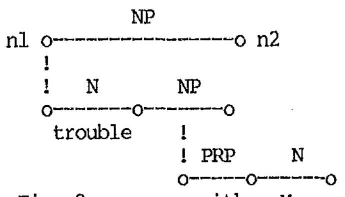


Fig. 2

If we interpret the nodes as arcs, we receive the S-graph representation (Fig. 2). Hence, we shall use 'node' and 'arc' as synonyms. The ambiguity of 'trouble with Max' is represented by a sequence of two NP-arcs that also goes from n1 to n2.

Much like most modern grammar theories (LFG, GPSG etc.), the CAP-concept is based on context-free rules. CAP differs from these theories in the way well-known problems of cf-grammars are dealt with. Where GPSG employs meta-rules, derived categories, and the ID/LP-formalism, LFG uses different structural concepts (C- and F-structures) and - above all - lexical knowledge. LFG and GPSG are augmented PS-grammars. With CAP PS-grammar has been abandoned. This is due to the fact that PS-grammars imply strict word order, and non-standard word order can only be handled by means of transformations (TG) or derived categories/new formalisms (GPSG).

In principle, in CAP constituents are accepted where they are found in natural language utterances. It is assumed more natural to accept and represent the constituents 'wen' (whom) and 'du' (you) in 'Wen liebst du?' (Who(m) do you love?) in their respective positions as accusative and nominative object than to mark the gaps in the representation where those constituents 'ought' to appear or to move them to their standard position and to leave a trace in the original place.

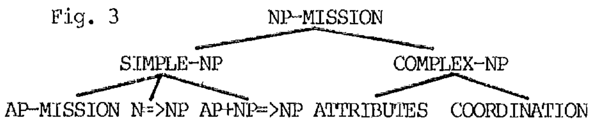
LFG and GPSG do not use transformations. In CAP transformations are possible, but they serve other purposes than in TG. They are not employed to account for structures that are not covered by standard PS-rules (the ID/LP-formalism was invented for that reason). On the one hand, transformations serve to 'normalise' certain surface structures, in order to make rule writing easier (cf. Luckhardt 1986). On the other hand, they produce the deep structure necessary for the disambiguation of lexemes and for other purposes of machine translation, e.g. by re-introducing deleted complements.

Unlike the government-and-binding theory, CAP moves constituents only in those cases, where this movement can be achieved without damage to representation without leaving a trace. E.g. in '... lastet dem Angeklagten das Verbrechen an.' (... charges the defendant with the crime.) the verbal prefix 'an' is moved to the left of 'lastet', so that the correct frame (i.e. the frame of 'anlasten' which requires a new dictionary look-up) can be used for assigning syntactic functions to the complements. TG-typical transformations like passive transformations are not employed, as the equivalent can be achieved by simple feature assignment.

In all, the CAP-parser for German (CAP-G) that is currently being developed may be regarded as a strictly controlled production system, where rule application is controlled in two respects:

a) 'missions' have to fulfil certain linguistic tasks. They are organised hierarchically, so that the higher missions may be said to be decomposed into partial (simpler) tasks (cf. Fig. 3). Thus the parsing strategy can be formulated quite explicitly. For every mission an 'expectation' may be formulated that allows it to select parts of the database that look 'promising' for the application of certain rules. The mode of application (see below) can be determined by the linguist.

Fig. 3



b) If a linguistic task cannot be subdivided any further, a 'scout', that represents such an elementary task, selects a path from the data structure offered, i.e. an unambiguous sequence of arcs, and tries to apply a rule or set of rules to this path. The grouping of rules into larger units has also been proposed by Carter-/Freiling 1984 and others.

This way of organising rules safeguards that the rule writer is relieved of looking at parallel structures. Rules can be simple, since feature agreement may be checked in missions and scouts so that rules may be kept general enough to be used in different places, i.e. in different scouts. The linguist can be quite sure his rules are applied the way he wants them to and to the structures intended. In fact, certain rules would be quite harmful, if they were allowed to operate on arbitrary structures. Rules ought to be perspicuous, but we think they cannot always be as simple as theoretical linguists would like them to be.

The application of cf-rules such as NP+PRED=>PRED may be subject to a number of restrictions. Earlier experience with SUSY has shown that valency grammar (cf. below) is a good basis for such a strategy, e.g.:

```
     PRED + NP1 => PRED (NP1) / condition:
                              NP1 may fill a slot
                          in the valency frame of PRED
```

After the application of such rules the corresponding valency is deleted; these rules are applied in parallel and by iteration. They are based on what Dowty (cf. Dowty 1982) calls the 'grammatical relations principle'.

CAP rules are augmented, i.e. they are not just structure-building rules like the ones above, but contain also conditions for their application, formulated for the left-hand side, and assignments to the symbols on the right-hand side (see below). This approach, of course, is not new and has been taken in METAL, PATR-II, LIFER, DIAGRAM, and many other systems. The way conditions and assignments are formulated is described below.

CAP possesses strong lexical and morphological components. These stem from its predecessor and are believed to be a prerequisite for efficient parsing rather than a part of the parsing theory.

Dependency grammar offers a secure foundation for the analysis of free-word-order languages like German or Russian and by no means impedes the analysis of languages like English or French, as has already been demonstrated with the SUSY MT system in the 70's (cf. Luckhardt/Maas/Thiel 1984). Moreover, for the sake of easier rule writing, it is helpful to represent all arguments of a predicate as sister nodes of each other and as sister nodes of the predicate's governor. This approach supports frame-oriented linguistic procedures (e.g. for the analysis of complements and complement clauses, translation of valency-bound constituents etc.) in a direct way, whereas the representation of such phenomena is not so natural in a phrase structure notation.

Rules, scouts, and missions

CAP rules, scouts, and missions are written in a functional metalanguage (FUSL, cf. Bauer et al. 1986). There are five types of rules according to the effect they have:

blending rule:	A + B	=>	C
start rule:	A	=>	X (A)
right expansion:	A (X) + B	=>	A (X + B)
left expansion:	A + B (X)	=>	B (A + X)
concatenation:	A + B	=>	X (A + B)

A blending rule may be employed where a constituent structure does not have to be preserved, as in:

```
AUX+PTC => FIV     for: 'was' + 'treated' =>
treat(TENSE=PAST, MS=FINITE_VERB, VOICE=PASS)

AUX + INF => FIV     for: 'will' + 'treat' =>
treat  (TENSE=FUT etc.)
```

```
                            C
                     ---------------
                     !             !
                     !             !
                     !             !
o------o------o  =>  o------o------o
  A       B            A       B          Fig. 5
```

The assignment part of such rules, of course, has to furnish the new arc on the right-hand side with the respective property/value pairs (cf. brackets). The effect of A + B => C is demonstrated in Fig. 5.

The arcs A and B remain intact and may be used by other rules. Thus a quasi-parallel processing is guaranteed. In cases of non-ambiguous structures, A and B may be deleted explicitly in the scout that invokes the rule.

```
                  ----------
                  ! !  X   !
                  ! !      !
                  ! o----o !
                  !   A'   !
o--------o   =>   o--------o
    A                 A             Fig. 6
```

A start rule is employed where a non-terminal arc is constructed from a terminal. A => X (A) means that a new arc X is produced with A as its substructure which spans the same part of the data structure as does A, cf. Fig. 6.

An expansion rule adds an arc as a sister arc to the substructure X of another arc. A (X) + B => A (X + B) has as a result the structure represented in Fig. 7.

Fig. 7

```
                                  ----------------------
                              ! !        A            !
                              ! !                     !
                              ! o------o--------o     !
                              !    X        B         !
o----------o----------o  =>   o-------------o---------o
 !   A       B                  !   A           B
 !                              !
 o----------o                   o---------o
       X                             X
```

A + B (X) => B (A + X) is employed analogously.

Concatenation rules are used to express coordination:

```
NP + COMMA + NP = NEWNP (NP + COMMA + NP)
NP + CONJ  + NP = NEWNP (NP + CONJ  + NP)
```

These rules produce deep structures. For 'Peter, Mary and John' the structure in Fig. 8 is generated.

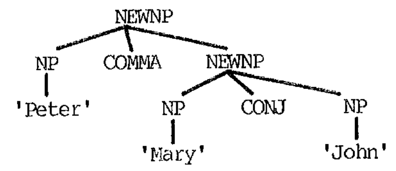


Fig. 8

CAP rules have the architecture given in Fig. 9.

```
rule  RULENAME
lhs             <left-hand side>
conditions      <restrictions on lhs>
rhs             <right-hand side>
assignments     <assignments to rhs>
end                                        Fig. 9
```

The conditions part may be empty. It allows navigation in the processed subchart and a variety of restrictions by means of logical expressions. This is also true for the assignments part, which, however, must be non-empty. An example is given in Fig. 9a.

```
rule PRED+SUBJ                        Fig. 9a
lhs   X + Y
conditions   eq   (MS of X, PRED)
             eq   (MS of Y, NP)
             notempty (int (FRAME of X,
                            SCASE of Y,
                            <NOM>))
             notempty (int (PERNUM of X,
                            PERNUM of Y))
rhs   Z ( subX + Y )
assignments   copydec (Z, X)
              assign  (SF of Y', SUBJECT)
              assign (FRAME of X,
                        min (FRAME of X, <NOM>)
              assign  (SCASE of Y, <NOM>)
end
```

Two neighbouring arcs X and Y are expected, X being a PRED, Y an NP. The FRAME of X is to include NOMinative, which also has to be one of the cases of Y. The PERNUM feature structures for person and number have to agree. The newly created arc Z that covers the substructure of X plus the nounphrase Y inherits all property/value-pairs from X. The (surface-)syntactic function SUBJECT is assigned to the new arc Y' which is a copy of Y. The NOMinative-slot is deleted from the FRAME of X. Y is given the unambiguous surface case NOMinative.

The system of missions and scouts guarantees that PRED+SUBJ is invoked, when the chart consists of PREDs and NPs, i.e. when the SIMPLE-STRUCTURES-mission has turned terminal elements into simple nonterminal ones (e.g. FIV=>PRED, DET+N=>NP etc.). By iteration, the output of PRED+SUBJ is used to attach the rest of the complements (by rules like PRED+DAT, PRED+PRPOBJ,AKK+PREDetc.).

Rules are grouped under and activated by what we call 'scouts'. A scout selects those paths (= unambiguous sequences of arcs) from the S-graph to which the rules of the scout may be applied. The modes of application are:

parallel: all rules are applied to the same structure
stratificational: one rule is applied after the other (stop after failure)
preferential: stop after success
iterative: repeat after success

The architecture of scouts is given in Fig. 10.

```
scout  SCOUTNAME
   conditions
            < path with conditions on arcs >
   use      rule RULE1
   ...
   use      rule RULEn
   params   <mode of application>
   options  <further options>
end                     Fig. 10
```

<path> is a sequence of normally not more than four arcs each of which is described in the <conditions on arcs> part (cf. Fig. 10a).

```
conditions                       Fig. 10a
  arc 1 (X , member (MS of X ,
             <ART-DEF,ART-INDEF,DEM,POSP,IND>))
  arc 2 (Y , equal  (MS of Y , N))
```

Here two neighbouring arcs X and Y are described, 'X' and 'Y' being names used only by this scout. The morphosyntactic category (MS) of X must be a member of the set in angled brackets, the MS of Y must equal N. The scout selects all sequences ART-DEF + N, ART-INDEF + N etc. one after the other from the database offered by a mission (see below) and tries to apply its rules to them. The angled brackets enclose the set of determiner types that are thought to be relevant here (def. art., indef. art., dem. pronoun, poss. pronoun, indef. pronoun) and that may be combined with a noun to form an NP. Other scouts select paths like PREP + N, PREP + AP + N etc. They all have to be dealt with in different scouts, as the conditions for unifying them into an NP and the values the NP's inherit are quite different.

Scouts are controlled by 'missions'. The system of rules, scouts, and missions presents the control structure of the parser (cf. example in Fig. 12). The elementary tasks of the parsing mission are organised as scouts that activate those (sets of) rules that are to be applied to fulfil the intended task. The linguists are free to choose the strategy they like according to the field they intend to cover. The modes of application are the same as above. The architecture of missions is given in Fig. 11.

```
    mission  MISSIONNAME                    Fig. 11
      expectations  left-context
                    scope <active area>
                    right-context
      subproblems solve (subproblem 1)
                  solve (...)
                  solve (subproblem n)
                  parameters
                  goal <goal structure>
    end
```

```
 mission PARSE-GERMAN                          Fig. 12
    mission SIMPLE-STRUCTURES
               scout N=>NP
                  rule N=>NP
               scout DET+ADJ+N=>NP
                  rule ARTD+ADJ+N
                  rule ARTI+ADJ+N
                  rule POSP+ADJ+N
    mission COMPLEX_STRUCTURES
        mission COMPLEX_NPS
           mission ATTRIBUTES
              mission GENITIVE_ATTRIBUTE
              ...
           ...
        ...
    ...
 end
```

A mission consists of a list of submissions or scouts that are applied in the mode <mode>, if certain 'expectations' (=preconditions) are fulfilled. The expectations part may be empty, so that the scouts may operate on the complete database. A well-defined structure may be formulated as the 'goal' of the mission. The expectations part describes a section of the S-graph where the scouts of that mission may be successful, i.e. this section with all its ambiguities (= parallel arcs) is taken from the database and handed over to the scouts. An example is given in Fig. 13.

```
 expectations                                 Fig. 13
   scope  first  (X , equal  (MS of X , FIV))
          mid    (Y , member (MS of Y , <NP,AP>))
          last   (Z , equal  (MS of Z , VERBPREFIX))
   right-context  (R , member (MS of R ,
                             <SEN,COMMA,NKO,SEM>))
```

The part of the database between the nodes n1 and n2 (cf. Fig. 14) is selected with all parallel structures, 'das Rauchen' being analysed as 'definite article + noun' (in one NP) and as 'personal pronoun + noun' (in two NP's). The expectation is to be read as follows: The first arc must be marked 'finite verb', the last one 'detached verbal prefix'. Between them one or more NP's and/or AP's (adjective phrases) in arbitrary distribution are expected. A full stop, comma, coordinating conjunction, or semicolon must be the right neighbour of Z, i.e. the arc left of n2. If these expectations are fulfilled, the partial S-graph that begins with X and ends with Z including all parallel arcs is activated for the scouts of that mission. These expectations are so explicit, because in this way structures may be disambiguated quite safely. In German, most verbal prefixes may also be prepositions, cf. (1) and (2).

(1) Er gibt das Rauchen <u>auf</u>.
(He gives up smoking.)
(2) Er gibt ein Konzert <u>auf</u> der Gitarre.
(He gives a concert on the guitar.)

The expectations described exactly fit for (1), but not for (2), and the mission activates the database accordingly.

```
            -------------------------
            !  !       NP          !
            !  o----o---------o    !
            !  ART       N         !
n1 o-------o-------o--------------o--------o n2
     gibt  ! das   !   Rauchen    !   auf
           !       !              !
            -------  -------------
            ! NP    !   NP
            !       !
            o----o  o--------o
Fig. 14     PRON        N
```

The scouts used for the analysis of detached verbal prefixes are the following:

```
solve  RIGHT-EXPANSION
solve  PRED+VZS
```

The first scout increments the predicate in the partial database between n1 and n2 until all NP's between the predicate and the verbal prefix are in the predicate's substructure, and the second scout concatenates verb and verbal prefix. The complete mission will look like Fig. 15.

A different approach to this problem is 'normalisation' mentioned above, where the verbal prefix is moved to the finite verb in the first place.

```
mission PARSE-VERBAL-PREFIXES:                 Fig. 15
  expectations
    scope  first  (X, equal  (MS of X, FIV)
           mid    (Y, member (MS of Y, <NP,AP>)
           last   (Z, equal  (MS of Z, VERBPREFIX)
      right-context (R, member (MS of R,
                           <SEN,COMMA,CONJ,SEM>))
  subproblems  solve (RIGHT-EXPANSION)
               solve (PRED+VZS)
               goal  (G, equal  (MS of G, PRED))
end
```

Feature propagation

When building syntactic structures, a parser transports features between nodes. In many modern grammar theories and formalisms this transport is achieved by unification (cf. Shieber 1985, Karttunen 1984, Kay 1984). For a number of reasons unification has no place in the CAP-concept (cf. Luckhardt 1986a). Unification was introduced as a simple instrument, which in fact has to achieve a very complex task. Feature propagation is too complex to be achieved by simple unification, and if the effect of unification is differentiated, it looses its elegance.

In a rule like 'DET+ADJ+N=>NP' it has to be stated which features are inherited by the NP, i.e. ADJ and N may have a feature FRAME, but only that of the N may be propagated. The same seems to be true for the semantic class.

A difference has to be made between selective (FRAME) and inherent features (CASE). Karttunen (1984) gives an example where by unifying 'I (CASE=NOM)' and 'do' the feature CASE=NOM is inherited by the new predicate 'I do (CASE=NOM)' which is not really desirable. There are more cases where unification leads to undesirable feature propagation.

Especially in coordination features have to be matched explicitly which, perhaps, is not so obvious for English. The structures in Fig. 16 (out of the house and across the street) have to be unified without PCASE and CASE having to match. In Fig. 17 (from the conduct of Eva and her husband), however, the CASE-values have to match, in order to prevent the coordination of 'aus dem Verhalten' and 'ihres Mannes', and PCASE=AUS is inherited by the new NP.

```
  aus dem Haus    und    über die Straße        Fig. 16
o--------------o------o------------------o
   PCASE=AUS             PCASE=ÜBER
   CASE=DAT              CASE=AKK

 aus dem Verhalten   und   ihres Mannes        Fig. 17
o-------------------o------o---------------o
! PCASE=AUS                  CASE=GEN
! CASE=DAT
!            Evas
o-------------------o
    CASE=GEN
    SF=GEN-ATTR
```

Only those features can be unified that are carried by at least one of the constituents, so that it is not easy to introduce features during the parsing mission, which is desirable in certain cases (cf. Luckhardt 1986a). On the other hand, it seems impossible to get rid of features that are no longer used, like the INFL-feature (after the agreement between the elements of an NP has been checked).

In CAP, the effect of unification is achieved by an operation that consists of a test and an action using FUSL-functions like

```
eq  (NUMBER of X, NUMBER of Y)
int (FRAME of X, SCASE of Y)
member (MS of X, <ARTD,ARTI,POSP,DEM,IND>
assign (SF of Y, SUBJECT)
```

Thus explicit comparison, creation, deletion, and propagation of features is possible.

Conclusion

CAP has to be seen in the context of automatic analysis and translation of natural language. It commands a formalism that makes it suitable for the development of efficient parsers by allowing for extensive means to represent linguistic knowledge and strategies for its use. The way these aspects interact is currently being formalised by Thiel in his NLPT (Natural Language Processing Theory, cf. Thiel 1985).

The underlying data structure is the S-graph, which allows the management of all kinds of ambiguities; moreover, the software system makes it unnecessary for the linguist/user explicitly to take care of ambiguities. Thus he/she may write rules without worrying about parallel structures, as his-/her view of the data structure is a simple tree or sequence of trees. There are methods, however, for indicating preference to certain structures over others.

Underlying linguistic features such as rule augmentation, feature propagation, lexicalisation etc. that are known from GPSG, FUG, LFG etc. have been extended to cover more phenomena, especially those encountered when parsing German. They are used in a way that allows the analysis of random samples of text in comparably short time.

Some special applications of CAP are

- normalisation: removal of idiosyncrasies and treatment of constructions that are notorious for the problems they present (discontinous verb forms, parentheses, etc.)
- formalisation of the complex agreement conditions on German NP's, treatment of free word order
- coping with complex forms of coordination
- controlled inheritance of features
- giving the linguist the opportunity of determining the grade of featurisation and the depth of representation

References

Bauer, M., Licher, V., Luckhardt, H.-D., Schäfer, Th., Schworm, C., Thiel, M. (1985). **FUSL - eine funktionale Sprache zur Repräsentation linguistischen Wissens und linguistischer Strategien.** Linguistische Arbeiten des SFB 100 Neue Folge, Heft 16. Saarbrücken: Universität des Saarlandes

Carter, A.W., Freiling, M.J. (1984). **Simplifying Deterministic Parsing.** In: Proceedings of Coling 1984, 239-242

Dowty, D. (1982). **Grammatical Relations and Montague Grammar.** In: P. Jacobson, G.K. Pullum (eds.). **The Nature of Syntactic Representation.** Dordrecht: Reidel

Karttunen, L. (1984). **Features and Values.** In: Proceedings of Coling 1984

Kay, M. (1977). **Morphological and syntactic analysis.** In: A. Zampolli (ed., 1977). **Linguistic Structures Processing.** Amsterdam: North-Holland

- (1984). **Functional Unification Grammar: a Formalism for Machine Translation.** In: Proceedings of Coling 1984

Luckhardt, H.-D. (1985). **Parsing with Controlled Active Procedures.** CL-Report No. 2. Saarbrücken: Universität des Saarlandes: SFB 100

- (1985a). **Valenz und Tiefenkasus in der Maschinellen Übersetzung.** CL-Report No. 4. Saarbrücken: Universität des Saarlandes: SFB 100

- (1985b). **Kontrollierte Mächtigkeit: Regeln in CAP.** CL-Report No. 8. Saarbrücken: Universität des Saarlandes: SFB 100

- (1986). **Normalisierung deutscher Oberflächenstrukturen mit controlled active procedures.** CL-Report 10. Saarbrücken: Universität des Saarlandes: SFB 100

- (1986a). **Vererbung von Merkmalen mit controlled active procedures.** CL-Report No 11. Saarbrücken:

Universität des Saarlandes: SFB 100

Luckhardt, H.-D., Maas, H.-D., Thiel, M. (1984). **The SUSY-E Machine Translation System.** Working Paper. Saarbrücken: Universität des Saarlandes: SFB100/A2

Maas, H.-D. (1984). **Struktur und Steuerung der linguistischen Prozesse in SUSY-II.** In: U. Klenk (ed., 1985). **Kontextfreie Syntaxen und verwandte Systeme.** Linguistische Arbeiten. Tübingen: Niemeyer

- (1985). **SUSY-II-Handbuch.** Linguistische Arbeiten des SFB 100 Neue Folge, Heft 14. Saarbrücken: Universität des Saarlandes

Shieber, St. M. (1985). **An Introduction to Unification-Based Approaches to Grammar.** Presented as a Tutorial Session at the 23rd Am. Meeting of the Ass. f. Comp. Ling., July 1985

Simmons, R.F. (1985). **Technologies for Machine Translation.** In: Proceedings of the Int. Symposium on MT, Tokyo, 14th Oct 1985

Thiel, M. (1985). **Eine konzeptionelle Basis für natürlichsprachliche Systeme.** Paper for the GLDV-Jahrestagung 1985, Hannover. Working Paper, Saarbrücken: Universität des Saarlandes

- (1985a, forthcoming). **Weighted Parsing.** In: L. Bolc (ed.). **Natural Language Parsing Systems.**

Varile, N. (1983). **Charts: A Data Structure for Parsing.** In: M. King (ed.). **Parsing Natural Language.** London: Academic

Winograd, T. (1983). **Language as a Cognitive Process.** Reading, Mass.: Addison-Wesley P.C.