[*International Conference on the State of Machine Translation in America, Asia and Europe*. Proceedings of IAI-MT86, 20-22 August 1986, Bürgerhaus, Dudweiler]

A CAP parser generator for German

Heinz-Dirk Luckhardt
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 solved 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.

The present paper introduces a CAP parser generator for German and gives an example. The term 'parser generator' is used to mean 'a software environment for the creation of parsers for specific purposes out of a given set of rules, scouts, and missions'.

Introduction of 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 morphosyntactic type (MS, i.e. the terminal or non-terminal category), a surface-syntactic function (SF), a deep-syntactic function (DSF), a semantic relation (SR), a weight, and information specific to an MS.

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

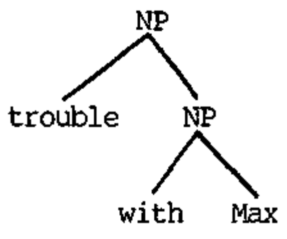


Fig. 1

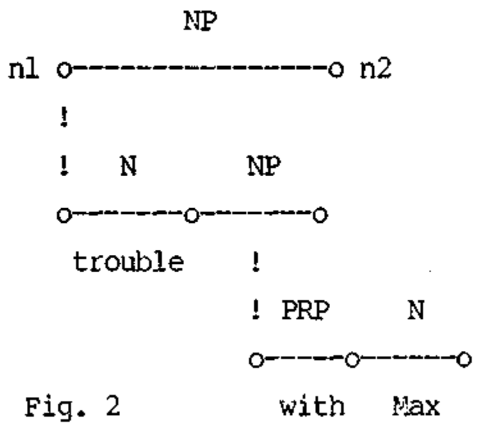
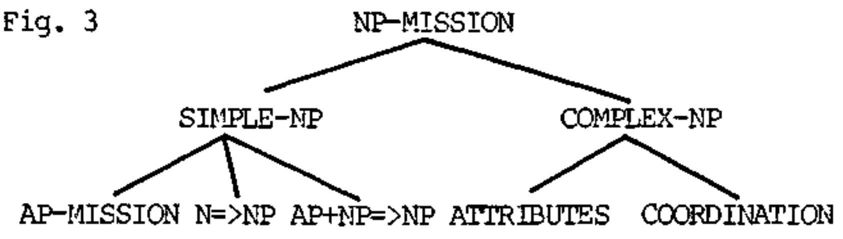


Fig. 2

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.

Fig. 3



In all, CAP-parsers may be regarded as strictly controlled production systems, 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.

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.

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 by missions and scouts so that rules may be kept general enough to be used in different places, i.e. by 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 <u>valency grammar</u> (cf. below) is a good basis for such a strategy, e.g.:

```
PRED + NP1 => PRED  (NP1) /   condition: NP1   fills 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.

CAP rules are augmented, i.e. they are not just structure-building rules, 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 <u>prere-</u>

guisite 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.)
```

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.

```
                          C
                     -----------
                     !          !

                     !          !
                     !          !
o--- o---- o =>      o----o --- o
  A    B                A     B         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 by 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.

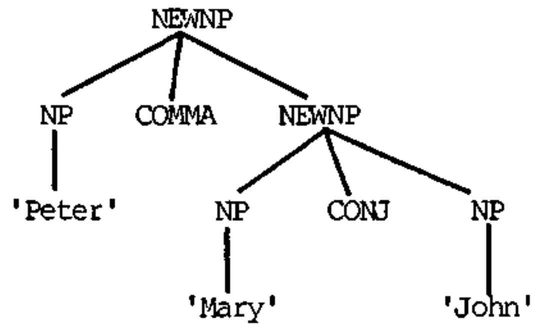


Fig. 8

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. 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.

```
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
```

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 non-terminal 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+PRED etc.).

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

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 a scout commands 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.

<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 by 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 <mode>
                    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 to the righthand side 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 auf.
  (He gives up smoking.)

(2) Er gibt ein Konzert auf der Gitarre.
  (He gives a concert on the guitar.)

Fig.14

```
          ------------------------------------
          !  !     NP          !
          !  o----o------o     !
          !  ART     N         !
nl o ----o ----o -------------o-----o n2
    gibt ! das !  Rauchen     ! auf
          !     !              !
          --------------------
            ! NP  !  NP
            !     !
            o---o o------o

            PRON     N
```

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

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 nl 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 loses 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, cf. Luckhardt 1986a).

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.

A concrete CAP-implementation

The CAP-concept and the background software allow the generation of parsers for specific purposes, e.g. a parser for noun phrases or for simple main clauses by creating new missions out of the set of existing missions, scouts, and rules. In the following, a general-purpose CAP-parser for German will be described. It commands

150 augmented cf-rules
 74 scouts
 40 missions

These have been implemented in the SUSY-II formalism in the past few years and are currently being transferred into the FUSL-formalism. Thus the performance of the parser can only be given for the SUSY-II implementation. The parsing speed is about one word per second CPU-time.

The lexical background is the Saarbrücker Deutsches Analysewörterbuch SADAW (145.000 entries). The input chart with the morphosyntactic descriptions of the terminal elements is produced by the morphological component of the SUSY MT system. It may be claimed that this morphoanalytic module is capable of producing a morphosyntactic description of any German input word, however complex it may be, with a very low error rate.

I shall only discuss those rules, scouts, and missions that take part in the analysis of

(3) Eine Frau bat ihren Mann darum, die Tür zu öffnen.
    (A woman asked her husband to open the door)

A similar sentence has been used in the demonstration of the Stuttgart LFG-implementation (cf. Frey/Reyle 1983) and may be used as the basis of comparison:

(4) A woman expects an American to win.

For (3) the following cf-rules are used:

```
DET + N        => NP
ADV            => NP
FIV            => PRED
TO + INF       => PRED
PRED + NP      => PRED
NP + PRED      => PRED
PRED + PRED  => PRED
PRED + COMMA => PRED
```

How and to which edges they are applied and how they are augmented will be dealt with in the following.

I shall describe the parse in a top-down fashion, i.e. I shall start by giving the top mission and end by stating the rules. The complete control structure is given in Fig. 18.

The processing mode concerns the scouts/missions/rules immediately dominated by a mission/scout.

The initial chart for (3) produced by the SUSY morphological analysis and dictionary look-up is given in Fig. 19 (only the MS-values are represented, as the morphosyntactic information on the arcs would require too much space here).

```
mission  DEUTSCH
parallel |-mission  SIMPLE-STRUCTURES
         |  parallel  |-mission VERB-PHRASES
         |            |   |-scout ZU+INF
         |            |       └rule ZU+INF
         |            |___ scout ONE-WORD-NONTERMINALS
         |            |    preferential  -rule ADV-NP
         |            |                  -rule N-NP
         |            |                  -rule FIV-PRED
         |            |                  -rule INF-PRED
         |            └-mission   2-WORD-NONTERMINALS
         |                ↓scout DET+SUB
         |                        |-rule DET+SUB
         |-mission   COMPLEX-STRUCTURES
         | parallel    |-mission NOUN-PHRASES
         |             |-mission PREDICATES
         |                iterative     |
         |                parallel     |-scout  LEFT-EXPANSION
         |                             |  parallel |-rule SUBJ+PRED
         |                             |           └-rule OBJ+PRED
         |                             |-scout  RIGHT-EXPANSION
         |                             | parallel |-rule PRED+SUBJ
         |                             |          |-rule PRED+OBJ
         |                             |          └-rule PRED+PRPOBJ
         |                             |-scout  INF-COMPL-CLAUSE
         |                               preferential |-rule CORR-CLAUSE
         |                                            |-rule LEFT-CLAUSE
         |                                            |-rule RIGHT-CLAUSE
         └-mission  DEEP-STRUCTURES
           sequential |-mission  DELETION-IN-COORDINATION
                      |-mission  PASSIVES
                      |-mission  SUBJECT-FOR-INFINITIVE-CLAUSE
                          └──scout  RESTORE-SUBJ
Fig. 18                             └── rule RESTORE-SUBJ
```

```
 Eine Frau bat ihren Mann darum , die Tür zu öffnen .

o----o----o---o-----o----o-----o-o---o---o----o------o—o

ARTU    N  FIV  POSP  N    ADV  !ARTB! N ! ZU !  INF  !
                                -----     ----  ------
                                !REL !    PRP   FIV
Fig. 19                          --
                                PER
```

Parsing starts with SIMPLE-STRUCTURES, where scouts select paths with one to three arcs to which constituent structure rules are applied. They produce the intermediate structure in Fig. 20.

```
                                          NP
                                    -------------
Fig.  20                            ! !          !
                                    !o----o---o  !
                                    !ARTB N      !
                                    !            !
---------- ----  ------------- ------  ------ ---- -------------
!  NP     !PRED!       NP     !  NP ! ! NP  ! NP !    PRED      !
!         !    !              !     ! !     !    !              !
!Eine Frau! bat!  ihren   Mann!darum!,! die ! Tür!  zu   öffnen !.
o----o----o----o--------o-----o-----o-o-----o----o-----o--------o
 ARTU!  N !FIV    POSP  !  N  ! ADV   ! REL ! N  ! ZU  !   INF  !
     -----              ------         -----     ------!        !
     ! NP               !  NP          ! NP         PRP ---------
     !                  !              !               !  PRED
      o---o             o-----o        o-----o         !
        N                 N              PER           o--------o
                                                          FIV
```

Parsing proceeds with COMPLEX-STRUCTURES. There are no complex NPs in our example, so NOUN-PHRASES will not produce any new structures. LEFT-EXPANSION and RIGHT-EXPANSION will change the data structure significantly (cf. Fig. 21).

Fig. 21

```
 o ---------------- o --- o ------------------------ o
 !        PRED      COMMA             !  PRED
 !                                    !
 ! NP     FIV     NP     NP           !  NP      ZU+INF
 o ---- -o------o --- o----------- o  o-------o ---- ----o
 !SF=SUBJ SF=GOV!SF=OBJ  !SF=OBJ-UM  !SF=OBJ  SF=GOV
 !      'bat'   !        !            !       'öffnen'
  o----o----o   o----o---o o---o      o---o---o
   eine Frau    ihren Mann darum       die Tür
```

This is only an excerpt from the S-graph at this stage with all unwanted structures omitted and our interest focussed on those parts that will eventually be used for the correct result. This leaves us with two problems:

1. to produce a PRED-arc that spans the whole chart
2. to produce a subject for the infinitive clause

For the first problem we employ the rule

PRED1 + PRED2 = PRED3

with - theoretically speaking - two interpretations:

PRED1 + PRED2 = PRED3 (substructure PREDl + PRED2)
PRED1 + PRED2 = PRED3 (PREDl + substructure PRED2)

depending on whether the infinitive clause is PRED2 or PREDl. The parsing of (3) complicated by the presence of the correlate 'dar-um' that serves as a substitute for the complement clause (cf. chart in Fig. 21). (3) would be as correct without it and would then be parsed by RIGHT-CLAUSE:

(3') Eine Frau bat ihren Mann, die Tür zu öffnen.

The rule for (3) has to consider the fact that the slot for the infinitive clause in the valency frame of 'bat' has already been filled and that the slot filler must be replaced by the clause. The lexicon entry for 'bat' (and all other forms of 'bitten') has to carry the following features:

```
bitten: VERB
          NP-frame : <subject, object, um-object>
          clause-frame: <um-object>
          clause = infinitive clause
          correlate = optional
          subject of clause = object
```

The rule CORR+CLAUSE looks as follows (it has been simplified insofar, as the comma has been left out of consideration):

```
rule CORR+CLAUSE
lhs P1...KOR) + P2
conditions eq (SPECIAL_FEATURE of KOR, CORRELATE)
           eq (INVENTORY of P1, ZU+INF)
           int (CLAUSE-FRAME of P1, SF of KOR)
rhs P3 (substructurePl + P2)
assignments assign (SF of P2', SF of KOR)
            assign (SF of KOR, CORRELATE)
end
```

Fig. 22

This turns the chart into that in Fig. 23.

```
o -----------------------------------o
  !      PRED
  !
  !  NP    FIV      NP        NP        PRED
  o-- --o-------o------o----------o----------- o
   SUBJ   GOV     OBJ    CORRELATE!  0BJ-UM
                              !
                              !  NP   ZU+INF
                              o-----o ---- o
                                OBJ   GOV
```

Fig. 23

After the deep syntactic functions have been assigned, the rule RESTORE-SUBJ will copy the correct deep subject into the infinitive clause:

```
    rule RESTORE-SUBJ
    lhs P1 (P2)
    conditions eq (INVENTORY of P2, ZU+INF)
               not (son (P2, X, eq (DSF of X, DSUBJ)))
               eq (CLAUSE_SUBJECT of P1, DOBJ)
    rhs P1 (P2(SLOT + substructureP2))
    assignments cond (son (P1, Y, eq (DSF of Y, DOBJ)),
                   copydec (SLOT, Y),
                   assign (DSF of SLOT, DSUBJ)
    end
```

Fig. 24

In the geometry the slot arc for the subject is prepared by the variable SLOT. This slot is filled in the assignments part by copying the arc with 'deep syntactic function = deep object (DSF = DOBJ)' under predicate P1, so that the final S-graph will look like Fig. 25.

```
 o ------------------- o
  !          PRED
  !
  !    NP      FIV     GN        NP      PRED
  o--------o---------o----------o----------o----------o
   !SF=SUBJ    SF=GOV !SF=OBJ    !SF=CORR   ! SF=OBJ-UM
   !DSF=DSUBJ 'bitten'!DSF=DOBJ  !          !
   o---o-----o        o----o---o  o-----o   !
   Eine Frau            ihr Mann    darum   o------o---o-----o
                                           !NP      !N      ZU+INF
                                           !SF=SUBJ !SF=OBJ SF=GOV
                                           !DSF=DSUBJ!DSF=DOBJ 'öffnen'
                                           !         !
                                           o---o----o o----o---o
                                            ihr Mann   die  Tür
```

Fig. 25

This chart may be represented by the functional structure in Fig. 26. An explanation of the category names will be helpful:

PRED	= predicate	<decor>	= decoration
DSUBJ	= deep subject	DEF-ART	= definite article
DOBJ	= deep object	POSS-PRON	= possessive pronoun
SSUBJ	= surface subject	OBJ-UM	= um-object
SOBJ	= surface object	FIV	= finite verb
GOV	= governor	IZU	= infinitive with 'zu'
KORR	= correlate		

```
PRED       (   DSUBJ <decor>  ( NP     DEF-ART      'eine' )
<decor>    (   SSUBJ          (          <decor>           )
           (                  (     GOV NOUN         'Frau')
           (                  (          <decor>           )
           (
           (  GOV   <decor>   [ FIV                'bitten']
           (
           (  DOBJ   <decor>  ( NP     POSS-PRON     'ihr' )
           (  SOBJ            (        <decor>             )
           (                  (     GOV   NOUN       'Mann')
           (                  (           <decor>          )
           (
           (  CORR  <decor>   [ NP  GOV ADV         'darum']
           (                  [           <decor>          ]
           (
           (  OBJ-UM   PRED   (DSUBJ  <decor> ( NP  POSS-PRON    'ihr' ) )
           (         <decor>  (SSUBJ          (     <decor>            ) )
           (                  (               (     GOV NOUN    'Mann' ) )
           (                  (               (        <decor>         ) )
           (                  (
           (                  (DOBJ   <decor> ( NP  ART-DEF       'die' ) )
           (                  (SOBJ   <decor> (      <decor>            ) )
           (                  (               (     GOV NOUN      'Tür' ) )
           (                  (               (        <decor>          ) )
           (                  (
           (                  (GOV    <decor> [ IZU            'öffnen' ] ) )
```

Fig. 26

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 (discontinuous 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/user 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

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: SFB 100/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.