UPPSALA UNIVERSITY

Project ref. no. *LE3-4239*

Project title *SCARRIE Scandinavian Proof-reading Tools*

Deliverable number *DEL 6.6.3*

Deliverable title *A grammar checking module for Swedish (RE)*

Number of pages *25*

WP/Task responsible *Anna Sågvall Hein, Department of Linguistics, Uppsala University, Box 527, S-751 20 Uppsala, Sweden*
E-mail: anna@ling.uu.se

Author(s) *Anna Sågvall Hein*

EC Project Officer *Antonio Sanfillipo*

Keywords *Grammar checking, partial parsing, chart parsing, local error rules, grammar error coverage, Swedish*

Abstract The grammar checking module of Swedish SCARRIE is described. It is based on partial parsing and the application of local error rules. The strategy has been implemented by means of ScarCheck, a technology that was developed in the project with the Uppsala Chart Processor, UCP, as its basis. ScarCheck, has two basic components, UCP, and a chart scanner, ReportChart. The parser, basically, uses a bottom-up strategy. It builds as much structure as the grammar allows. Errors are recorded in the chart, and the scanner traverses the chart in search for errors to be reported. The report includes a presentation of the technology and the basic operation of the checker, the UCP formalism, the grammar and its error coverage, and a discussion of the potential for further development. Swedish SCARRIE is available for testing at http://stp.ling.uu.se/~ljo/scarrie-pub/scarrie.html.

Executive Summary

The report focuses on two major aspects of the grammar checker of Swedish SCARRIE, the technology in which it was implemented, ScarCheck, and the Swedish grammar and its coverage. The efforts devoted to the development of the technology were no less than those devoted to concrete grammar writing, rather the other way around.

The Swedish checker uses the ScarCheck technology. It was developed in the SCARRIE project with the Uppsala Chart Processor, UCP (Carlsson 1981, Sågvall Hein 1983), as its basis. It applies a combined approach to grammar checking based on robust partial parsing, and the application of local error rules. Phrase constituents are analysed by means of robust rules that accept feature violation. Local error rules handle structural errors at clause and sentence level. The error rules may operate on the results of the parsing process. They are formulated in terms of phrase categories, basic syntactic categories, lemmas, and morpho-syntactic features. All the rules are integrated in one grammar. Typically, a grammar rule covers both the correct and the incorrect case.

The grammar focuses on a selected set of error types. It can be thought of as a phrase structure grammar implemented in an augmented state transition network style. Error features are inserted into the chart during rule application.

The fundamental problem when working with a grammar checker based on partial parsing is to avoid false alarms due to lexical ambiguity and false segmentation of the sentence. Erroneous constructions that are captured by the grammar may coincide with correct ones. The general strategy provided by ScarCheck to handle these problems is based on a grammar that generates both the correct and the incorrect analysis. If both analyses are recorded in the chart, the correct analysis "neutralises" or "covers" the erroneous one.

The grammar may apply to correct as well as to in correct text. In both cases it will generate a partial parse of those constructions that are covered by the rules. That will be NPs, APs, ABs, PPs, and fragments of VPs, declarative clauses, WH-questions, relative clauses, and explicative clauses. The grammar covers roughly 30 error types according to the SCARRIE error typology (see Wedbjer Rambell 1998a). The selection of errors currently implemented in the grammar is based on an analysis of the errors in the Error Corpora Data-base and their frequencies (Wedbjer Rambell et al. 1998; Webjer Rambell 1998b).

Work on the Swedish grammar checker has demonstrated that partial parsing and the application of local error rules is a viable strategy for grammar checking. The linguistic limits of the approach were set in an earlier study (Wedbjer Rambell 1998b) with regard to a detailed error typology (Wedbjer Rambell 1998a) of more than 500 error types.

The study has further demonstrated that the chart-based ScarCheck implementation of the strategy provides a well functioning and appropriate technology for the purpose.

The current Swedish grammar covers a subset only of the error types that can be handled in this framework. An extension of the grammar to comprise a larger span of error types should proceed in accordance with the error analysis that was referred to above. Further, in parallel with the implementation of new error types, a systematic study of coinciding correct constructions to be included in the grammar should be made to cover false alarms.

A grammar checking module for Swedish

Anna Sågvall Hein
Department of Linguistics
Uppsala University
anna@ling.uu.se

1 Introduction

In this report we will focus on two major aspects of the grammar checker of Swedish SCARRIE, the technology in which it was implemented, ScarCheck, and the Swedish grammar and its coverage. The efforts devoted to the development of the technology were no less than those devoted to concrete grammar writing, rather the other way around.

The grammar checker of Swedish SCARRIE uses the ScarCheck technology. It was developed in the SCARRIE project with the Uppsala Chart Processor, UCP (Carlsson 1981, Sågvall Hein 1983), as its basis. The checker applies a strategy of partial parsing and the application of local error rules (Sågvall Hein 1998a). It has two basic components, the UCP parser and a chart scanner, ReportChart (Starbäck 1999). The parser builds as much structure as the grammar, including local error rules, allows. Grammar rules are formulated in the procedural UCP formalism. Errors are recorded in the chart, and the scanner traverses the chart in search for errors to be reported. Scarcheck has been integrated with the CORRie spell checking framework. For further information about the general architecture of Swedish Scarrie and the integration with the CORRie spell checker, see Sågvall Hein 1998c.

The grammar of Swedish SCARRIE covers roughly 30 error types according to the SCARRIE error typology (see Wedbjer Rambell 1998a). The individual error types will be discussed in the presentation of the grammar below. Finally, we will discuss the potential of the ScarCheck machinery for further development.

2 Basic strategy

The checker applies a combined approach to grammar checking based on robust partial parsing, and the application of local error rules. Phrase constituents are analysed by means of robust rules that accept feature violation. Local error rules handle structural errors at clause and sentence level. The error rules may operate on the results of the parsing process. They are formulated in terms of phrase categories, basic syntactic categories, lemmas, and morpho-syntactic features. All the rules are integrated in one grammar. Typically, a grammar rule covers both the correct and the incorrect case(s). The grammar can be thought of as a phrase structure grammar with rules implemented in a state transition style allowing for feature testing with unification as the basic, even though not only, operation.

The chart parser uses a bottom-up strategy. This is in accordance with the partial nature of the parsing process; in the general case, there will be no edge spanning the whole chart. The grammar is formulated in a procedural formalism, and the rules are invoked from the grammar (Sågvall Hein 1983). For instance, NP rules are triggered at the recognition of categories that may appear as NP introducers. Rules designed to catch erroneously deleted finite verbs are triggered at the recognition of categories that may introduce clauses.

Unification of feature structures is the basic operation for testing and assignment in the procedural formalism. Errors are recorded as features with values set in accordance with the error typology that was developed in the SCARRIE project.

The chart scanner, Reportchart, traverses the chart in search for errors. Starting at the first vertex of the chart, it inspects the feature structure of the longest inactive edge. Possible errors are recorded in the error protocol, before the attention of the scanner is directed to the edges going out from the final vertex of the current edge. The process goes on to the end of the chart.

In traversing the chart, the scanner applies three basic principles

1. the *simplicity principle*
2. the *longest-span principle*
3. the *right-before-wrong principle*

The simplicity principle implies that search for an error will be limited to the top-level of the feature structure of an edge; errors occurring at lower levels in the description thus have to be propagated. According to the second principle, the longest outgoing edge from a vertex will be followed, disregarding any edge of a shorter span. According to the third principle, no error will be reported if there is more than one edge of an equal span, and among them there is at least one without an error feature. These principles are of major importance to the grammar writer. She may use them to avoid false alarms by covering them by longer or parallel edges representing correct analyses.

3 UCP

The original version UCP is written in Lisp. It takes strings of characters as its input and is capable of handling dictionary search, morphological analysis, and syntactic analysis. In the SCARRIE context, the CORRie spell checker is responsible for word recognition. For an integration of the two modules, a modification of UCP was thus called for. The basic idea was to modify UCP in such a way that it would accept input from CORRie in terms of strings of grammatical codes. For this purpose a new Lisp version of UCP was written, UCP2 (Starbäck 1999) and successfully integrated with CORRie (Sågvall Hein 1998b). Finally, for efficiency and conformity purposes a simplified version of UCP2 was written in C (Weijnitz 1999) and plugged into the system. It is referred to as UCP3, or UCP light.

3.1 Original Lisp version

Below follows a brief intuitive presentation of the UCP formalism from the perspective of the user. For a more formal presentation, see Carlsson 1981.

In UCP being a chart parser processing proceeds task by task. According to the fundamental law of chart parsing, a task is formed when an active edge meets an inactive one; the rule associated with the active edge is applied to the feature structure of the inactive edge outgoing from its end vertex. Successful rule application amounts to assigning features to the feature structure of the active edge. Once the rule is satisfied, the feature structure of the active edge is stored in the chart with a new inactive edge, its feature structure. The root attribute of the feature structure of the active edge is denoted & and that of the inactive edge is denoted *. The UCP formalism consists of a number of operators. Some of them account for the insertion of new edges into the chart, and some of them are used for testing and assignment purposes. There are also a few meta operators that may be used for rule transparency and compactness.

General rule format

(define *name.of.resource*-entry *unit.to.be.defined*
#u *rule.body;*
#! *inactive.filter*;
#! *active.filter*;
)

The rule body is an obligatory part of a rule whereas the two filters are optional. It is a sequence (Boolean 'and') of operations formed by means of the UCP operators. The inactive filter imposes restrictions upon the feature structure of the inactive edge for a task to be generated, and the active filter does the same thing with regard to the active edge. The inactive filter specifies that or those attributes of the inactive edge that have to be present for the rule to form a task with the edge. The active filter specifies that or those attributes in the active edge that have/have not to be present for the edge to form a task with an inactive edge. For a summary of the UCP operators, see App. A. For a detailed illustration of the operation of UCP in grammar checking, see Sågvall Hein & Starbäck (1998).

3.2 UCP2 and UCP3

The basic difference between UCP on the one hand, and UCP2 and UCP3 on the other, is that the former takes strings of characters as its input, whereas the other two take strings of grammatical codes as their input. They all work together with ReportChart, implying that there are, as a matter of fact, three versions of ScarCheck. The original version is an independent Lisp system using a dictionary of its own. It was used for the first experiments with chart based grammar checking in SCARRIE (see Sågvall Hein 1998a). ScarCheck2 and Scarcheck3 use UCP2 and UCP3, respectively. Basically, they provide the same functionality and they have both been integrated with the CORRie spell checker. ScarCheck2 provides richer tracing facilities and is primarily used for development and testing purposes, whereas the lighter C-version of is intended for the final version of the prototype. A web-based development interface for UCP3 was also implemented. It is to be found at url: http://stp.ling.uu.se/~ljo/scarrie-pub/ucp_light.html. See also App. B. In illustrating the operation of ScarCheck below, we will use the UCP2 Lisp version, i.e. ScarCheck2.

4 ScarCheck

ScarCheck2 runs in an interactive mode on the Unix platform. It is invoked by means of the `scarcheck2` command.

We will start the presentation of ScarCheck by an example of how it handles grammar checking by mean of partial parsing with feature relaxation. Our example will be the correct np "sina skuggor" [its shadows] followed by the same example with a number agreement error "sin skuggor", pronoun in the singular, noun in the plural.

```
>(sp '((#(sina.VB VBAIM) #(sina.PS PSXP) )(#(skugga.NN NNUPIB) )))
```

Since there is no error, no message will be given. We may however want to inspect the chart which can be done by means of the `show` command.

```
>(show)
```

```
1|      2|         3|
 .SINA.VB.SKUGGA.NN.
 .SINA.PS.-NP-------.
 .-NP---------------.
1|      2|         3|
```

The graphical representation of the chart shows its inactive edges only. Individual edges, active as well as inactive, may be inspected by means of the type command. It takes three arguments: initial vertex, final vertex, and, optionally, 't' to denote that only inactive edges should be displayed.

```
> (type chart 1 3 t)

1--3     Creator: 20
         Features: (* = (START = 1
                         END = 3
                         NUMB = PLUR
                         GENDER = NIL
                         POSS = (WORD.CAT = PS
                                 LEM = SIN.PS)
                         PHR.CAT = NP
                         HEAD = (LEM = SKUGGA.NN)))
```

In the first vertex we find the active edges that carry the rules that were initiated from the from the grammar (`start.rule`), and from the dictionary (NP_POSS, VPI) and, respectively.

```
> (type chart 1 1)

1--1     Creator: 2
         LR-Action: NP_POSS;
1--1     Creator: 2
         LR-Action: VPI;
1--1     Creator: 0
         LR-Action: START.RULE;
```

NP_POSS is initiated by the possessive pronoun *sina*, and VPI is initiated by the homograph verb. The first rule will eventually lead to the recognition of the NP whereas the VPI rule responsible for the recognition of infinitive verb phrases won't give any result. For an illustration of the grammar rule format, we present the NP_POSS rule:

```
(define sve.gram-entry np_poss
 #u  <& phr.cat> :=: 'np,
     (<* phr.cat> = 'np,
     <* case> = 'gen,
     <& def> :=: 'def,
     <& poss phr.cat>:=:<* phr.cat>,
     <& poss lem>:=:<* head lem>,
     /<* word.cat>='PS,
     <& numb>:=:<* numb>,
     <& gender>:=:<* gender>,
     <& poss word.cat> :=:<* word.cat>,
     <& poss lem> :=:<* lem>),
     advance(np.poss_np);

#!   phr.cat or word.cat;
)
```

The second alternative of the disjunction applies, and an advancement to the NP.POSS_NP rule is made. Below we display the active edge carrying this rule and the rule itself:

```
1--2     Creator: 10
         Features: (& = (NUMB = PLUR
                         GENDER = NIL
                         POSS = (WORD.CAT = PS
                                 LEM = SINA.PS)
                         PHR.CAT = NP))
        LR-Action: NP.POSS_NP;
```

```
(define sve.gram-entry np.poss_np
#u   <* phr.cat> = 'np,
     (<& numb>:=:<* numb>/assign.err('gpnpag01)),
     (<* head form>:=:'indef/assign.err('gpnpss02)),
     <& head lem>:=:<* head lem>,
     store,
     advance(np.coord.conj);

#!   phr.cat;
)
```

The NP.POSS_NP rule is designed to capture two kinds of errors, i.e. disagreement of the number category ('gpnpag01) and wrong species of the head noun ('gpnpss02). We will illustrate these two cases, starting with the number disagreement case "sin skuggor", followed by the wrong species case "sina skuggorna".

```
(sp '((#(sin.PS PSUS) )(#(skugga.NN NNUPIB) )))
INTERVALL: 1,2
FEL: gpnpag01: fel numerus [wrong number]

1|     2|         3|
 .SIN.PS.SKUGGOR.NN.
 .-NP--------------.
        .-NP-------.
1|     2|         3|

1--3     Creator: 12
         Features: (* = (START = 1
                         END = 3
                         NUMB = SING
                         GENDER = UTR
                         POSS = (WORD.CAT = PS
                                 LEM = SIN.PS)
                         PHR.CAT = NP
                         ERR = (1 = GPNPAG01)
                         HEAD = (LEM = SKUGGA.NN)))
```

The pronoun being in the singular number disagrees with the noun being in the plural as indicated by the error feature. The error codes follow the SCARRRIE error typology, and they are connected to a set of interchangeable error messages. User validation has to tell how to formulate the error messages in accordance with the user needs.

```
(sp '((#(sina.VB VBAIM) #(sina.PS PSXP) )(#(skugga.NN NNUPDB) )))
INTERVALL: 1,2
FEL:     gpnpss02: fel species [wrong species]

1|      2|        3|
 .SINA.VB.SKUGGA.NN.
 .SINA.PS.-NP------.
 .-NP--------------.
1|      2|        3|


1--3     Creator: 20
         Features: (* = (START = 1
                         END = 3
                         NUMB = PLUR
                         GENDER = NIL
                         POSS = (WORD.CAT = PS
                                 LEM = SIN.PS)
                         PHR.CAT = NP
                         ERR = (1 = GPNPSS02)
                         HEAD = (LEM = SKUGGA.NN)))
```

Two cases of grammar checking based on partial parsing with feature relaxation were demonstrated above. The applied strategy is quite simple. However, the abundance of lexical ambiguities that have to be considered when a dictionary of a realistic size is used makes grammar writing a challenge. Lexical ambiguities may cause structural ambiguities as well as the false identification of phrases crossing phrase boundaries. For an illustration of the ambiguity problem we use once again the number disagreement example, this time in the context of a sentence: "Natten bär sin skuggor." [The night carries its shadows.]

```
(sp '((#(natt.NN NNUSDB) )(#(bär.NN NNNXIB) #(bära.VB VBAPMI) )(#(sin.PS
PSUS) )(#(skugga.NN NNUPIB) )(PUNC)))
INTERVALL: 1,4
FEL: gpnpag01: fel numerus
```

The one and only error in the sentence is properly recognised and reported. The underlying chart, however, shows that quite a few partial analyses have been generated, in addition to the longest edge from 1 to 5 carrying the error.

```
1|           2|      3|     4|        5|      6|
 .-NATT.NN----.-BÄR.NN.SIN.PS.SKUGGA.NN.STOP.SR.
 .-NP---------.BÄRA.VB.-NP-------------.
 .CL.DECL.FRAG.-NP----.      .-NP------.
 .-NP-----------------.
 .-CL.DECL.FRAG-------.
 .-CL.DECL.FRAG------------------------.
              .VP.FRAG.
              .-VP.FRAG----------------.
1|           2|      3|     4|        5|      6|
```

Among the partial analyses we find five NPs, the correct ones: "natt" [night], "bär" [berry or berries], "skuggor" [shadows], and the erroneous ones: "sin skuggor" [its shadows] and "natten bär" [night berry/berries]. The number error in "sin skuggor" analysis has already been discussed. The "natten bär" though is due to an NP rule, designed to capture cases where an intended genitive attribute erroneously appears in the basic case. The error type is denoted

'gpnpca01 and was highly ranked in the analysis of the errors in the Swedish error data base (Wedbjer Rambell et al. 1998; Wedbjer Rambell 1998b). The error edge is displayed below:

```
1--3     Creator: 34
         Features: (* = (START = 1
                         END = 3
                         PHR.CAT = NP
                         NUMB = SING
                         GENDER = UTR
                         CASE = BASIC
                         DEF = DEF
                         HEAD = (FORM = <* DEF>
                                  WORD.CAT = NOUN
                                  GENDER = <* GENDER>
                                  LEM = NATT.NN)
                         ERR = (1 = GPNPCA01)
                         SECOND = BÄR.NN))
```

An analysis of "natten bär" as a case of 'gpnpac01 may not seem very natural but formally it is quite in order. A more natural analysis of the string is as a VP fragment, consisting of a noun followed by a finite verb. The alternatives are due to the ambiguity of "bär" denoting a noun or a finite verb. According to the right-is-better-than wrong strategy, ReportChart will disregard the erroneous NP edge from 1 to 3 in the chart since it is paralleled with a correct CL.DECL.FRAG edge embedding the VP analysis:

```
1--3     Creator: 36
         Features: (* = (START = 1
                         END = 3
                         PHR.CAT = CL.DECL.FRAG
                         INIT =
                         SUBJ = (START = 1
                                  END = 2
                                  PHR.CAT = NP
                                  NUMB = SING
                                  GENDER = UTR
                                  CASE = BASIC
                                  DEF = DEF
                                  HEAD = (FORM = <* SUBJ DEF>
                                           WORD.CAT = NOUN
                                           GENDER = <* SUBJ GENDER>
                                           LEM = NATT.NN))
                         VP = (LEM = BÄRA.VB)))
```

This edge is generated by a local error rule CL.DECL_NP (and its follow-up rule CL.NP_VFIN). It applies to declarative sentences with an NP in the first, foundation, position, and it is triggered from the grammar at the recognition of sentence initial NPs.

```
1--1     Creator: 4
         LR-Action: CL.DECL_NP;
1--1     Creator: 2
         LR-Action: NP_NOUN;
1--1     Creator: 0
         LR-Action: START.RULE;
```

The rule recognises an initial NP followed by a VP. There are two alternative VPs and both will be considered. Following the longest match principle, ReportChart will choose the

longest one from 1 to 5, and any erroneous edges of a shorter or equally long span will be disregarded:

```
1--5     Creator: 78
         Features: (* = (START = 1
                         END = 5
                         PHR.CAT = CL.DECL.FRAG
                         INIT =
                         SUBJ = (START = 1
                                 END = 2
                                 PHR.CAT = NP
                                 NUMB = SING
                                 GENDER = UTR
                                 CASE = BASIC
                                 DEF = DEF
                                 HEAD = (FORM = <* SUBJ DEF>
                                         WORD.CAT = NOUN
                                         GENDER = <* SUBJ GENDER>
                                         LEM = NATT.NN))
                         ERR = (1 = (1 = (1 = GPNPAG01)))
                         VP = (VERB = BÄRA.VB
                               NP = (START = 3
                                     END = 5
                                     NUMB = SING
                                     GENDER = UTR
                                     POSS = (WORD.CAT = PS
                                             LEM = SIN.PS)
                                     PHR.CAT = NP
                                     ERR = <* ERR 1 1>
                                     HEAD = (LEM = SKUGGA.NN)))))
```

Adhering to the simplicity strategy of ReportChart, the NP error was propagated to the top level of the feature structure.

Primarily, the declarative clause fragment rule (CL.DECL_NP and its follow-up rule CL.NP_VFIN) was designed as a local error rule to capture missing finite verbs in declarative main clauses. However, as should be clear from the example, it is also needed to cover up the analysis of "natten bär" as an NP with a case error. Without this rule and the positive VP rules that are used, the system would give a false alarm for the NP. From this example and others, we conclude that a proper grammar has to include quite a number of positive rules to balance the partial parsing rules with constraint relaxation. It is not a viable approach to restrict the grammar to those constructions that are targeted by the grammar checker only. In other words, even if the checker would focus on errors in the NP only, quite a few rules covering other ambiguous constructions would be needed. The same is true for errors captured by local error rules. They have to be balanced by positive rules focusing on structural ambiguities caused by lexical ambiguities. This is also the remedy against wrong segmentation otherwise causing false alarms. Most of the segmentation problems may be solved by taking a large enough context into account and formulating positive cover up rules. Below we present an example of such a case.

"Det älskade kvinnorna."

The sentence is either a full sentence with a pronoun object in the foundation position, a finite verb in the second position, and a subject in the final position [That the women loved.]. Or it may be a nominal phrase with a number and a gender agreement error [The beloved women.].

The two analyses are due to the ambiguous reading of "älskade" as a finite verb or as a past participle, and the relaxation of the NP rule. A positive sentence rule will cover up the analysis of the string as an erroneous NP.

```
(sp '((#(det.PN PNNSZ) #(det.AL ALNSD) )(#(älskad.PC PCPXSDBT)
)(#(kvinna.NN NNUPDB) )(PUNC)))

1|           2|        3|        4|      5|
 .-DET.PN-----.ÄLSKAD.PC.KVINNA.NN.STOP.SR.
 .-DET.AL-----.-VP.FRAG-.-NP------.
 .-NP---------.-ADJP----.
 .CL.DECL.FRAG.-VP.FRAG-----------.
 .-CL.DECL.FRAG---------.
 .-NP-------------------.
 .-CL.DECL.FRAG---------.
 .-CL.DECL.FRAG-------------------.
 .-NP-----------------------------.
              .-NP----------------.
1|           2|        3|        4|      5|

1--4     Creator: 148
         Features: (* = (START = 1
                         END = 4
                         PHR.CAT = NP
                         DET = (WORD.CAT = ART
                                LEM = DET.AL)
                         DEF = DEF
                         FORM = DEF
                         NUMB = SING
                         GENDER = NEUTR
                         ATTR = ÄLSKAD.PC
                         CASE = BASIC
                         ERR = (1 = GPNPAG01)
                         HEAD = (LEM = KVINNA.NN
                                 FORM = DEF)))

1--4     Creator: 128
         Features: (* = (START = 1
                         END = 4
                         PHR.CAT = CL.DECL.FRAG
                         INIT = +
                         SUBJ = (START = 1
                                 END = 2
                                 PHR.CAT = NP
                                 HEAD = (WORD.CAT = PRON
                                         LEM = DET.PN)
                                 NUMB = SING
                                 CASE = BASIC
                                 GENDER = NEUTR)
                         VP = (VERB = ÄLSKAD.PC
                               NP = (PHR.CAT = NP
                                     HEAD = (LEM = KVINNA.NN)))))
```

As shown in the display of the NP edge above, only a number agreement error (gpnpag01) has been recorded; no gender agreement error. This is in accordance with a decision that has been taken during the work on the grammar; one error type only is recorded on one and the same edge. It is up to the grammar writer to see to it that the most likely error is the one that is recorded. No weights are used. It is a matter of ordering the ucp operations in the rule.

Finally, we present an example of a structural error that is handled by means of a local error rule. It is, in fact, the same rule that is responsible for building sentence fragments of declarative sentences with an NP in the first position that was used for an illustration above. Thus local error rules may as well include means for recognising proper constructions. In this sense they are not any different from the partial parsing rules with constraint relaxation.

"Det nödvändigt att tänka i nya banor." [It necessary to think in new ways.]

```
((#(det.PN PNNSZ) #(det.AL ALNSD) )(#(nödvändig.AV AVNSIBP) )(#(att.IE IE)
)(#(tänka.VB VBAIM) )(#(i.PR PRP) )(#(ny.AV AVZZZBP) )(#(bana.NN NNUPIB)
)(PUNC) ))
INTERVALL: 1,2
FEL: gpvvmv01: predikatsverb saknas [finite verb missing]

1|          2|          3|     4|       5|   6|    7|      8|      9|
 .-DET.PN-----.NÖDVÄNDIG.AV.ATT.IE.TÄNKA.VB.I.PR.NY.AV.BANA.NN.STOP.SR.
 .-DET.AL-----.-ADVP-------.-IP------------.-PP--------------.
 .-NP---------.-ADJP-------.-IP-----------------.-ADJP.-NP----.
 .CL.DECL.FRAG.             .-IP-------------------------------.
 .-CL.DECL.FRAG------------.      .-VP.FRAG.    .-NP----------.
                                  .-VP.FRAG-----.
1|          2|          3|     4|       5|   6|    7|      8|      9|

1--3     Creator: 72
         Features: (* = (START = 1
                         END = 3
                         PHR.CAT = CL.DECL.FRAG
                         INIT = +
                         SUBJ = (START = 1
                                 END = 2
                                 PHR.CAT = NP
                                 HEAD = (WORD.CAT = PRON
                                          LEM = DET.PN)
                                 NUMB = SING
                                 CASE = BASIC
                                 GENDER = NEUTR)
                         VP = (COMPL = (START = 2
                                        END = 3
                                        PHR.CAT = ADJP
                                        ADJ1 = (WORD.CAT = ADJ
                                                 LEM = NÖDVÄNDIG.AV
                                                 DEGREE = POS)
                                        GENDER = NEUTR
                                        NUMB = SING
                                        CASE = BASIC
                                        PROP = NIL
                                        A-FORM = T
                                        FUNC = NIL
                                        FORM = INDEF))
                         ERR = (1 = GPVVMV01)))
```

5 The grammar

The grammar focuses on a selected set of error types and handles them by means of partial parsing and the application of local error rules as illustrated above. It can be thought of as a phrase structure grammar implemented in an augmented state transition network style. Error features are inserted into the chart in connection with rule application.

The grammar may apply to correct as well as to in correct text. In both cases it will generate a partial parse of those constructions that are covered by the rules. That will be NPs, APs, ABs, PPs, and fragments of VPs, declarative clauses, WH-questions, relative clauses, and explicative clauses. As regards declarative main clauses, inversion is taken into account. Local error rules are formulated to cover missing units of various kinds, doubled units of various kinds, word order problems, some valency problems, some graphical problems, and some cases of split words. For a rough outline of the contents of the grammar in terms of rule names, see Appendix B.

The selection of errors currently implemented in the grammar is based on an analysis of the errors in the Error Corpora Database and their frequencies (Wedbjer Rambell et al. 1998; Webjer Rambell 1998b). The grammar errors in the ECD were analysed and categorised into three groups: (1) errors that can be handled by partial parsing, (2) errors that can be handled by local error rules, and (3) errors that lie outside the scope of partial parsing and local error rules. The following order of priority was proposed as a result of the error analysis:

Primary grammar problems:
- agreement within the noun phrase
- exceptions from agreement rules (species)
- case problems
- verb sequences
- structural errors: violations of category sequences of well-formed phrases and clauses

Secondary grammar problems:
- verb valency
- agreement between NP (subject) and AP (subjective complement)
- noun valency
- adjective valency
- pronoun case

The order of priority presented above has served as a starting point in the implementation of the error types to be covered by the grammar. Examples of all the problems referred to as primary grammar problems are included. In addition, errors in the adjectival phrase were treated as a problem in its own right. From the secondary grammar problems category a few verb valency problems were implemented, i.e. agreement between NP and AP, verb valency concerning the use of the infinitive marker "att", and the pronoun case after preposition. In addition, the implemented grammar comprises rules for the recognition of missing second members of compound conjunctions, missing final parentheses, and split proper noun compounds. The problem areas listed above are quite broad, and not all the error varieties included in them are covered. Below we will go into some detail regarding the contexts that are handled.

Non-structural error types are handled by means of partial parsing with feature relaxation, so-called robust rules. Roughly, this category includes agreement within the noun phrase, exceptions from agreement rules (species) within the nominal phrase, case problems within the nominal phrase, agreement within the adjectival phrase, pronoun case after

preposition, agreement between the subject and the subjective complement. Structural problems including word order problems, missing units, doubled units, and noun valency problems are handled by means of local error rules. As regards errors in the verb sequence, some of them are treated by means of feature relaxation and some by means of local error rules. For at detailed account of how the totality of the error types comprised by the error typology are categorised with regard to how they may be treated (partial parsing, local error rules, outside the scope of these two strategies), see Wedbjer Rambell 1998b.

Below we present those error types that are currently implemented in the Swedish grammar. They constitute only a subset of the types that may be handled by means of the ScarCheck strategy. Continued work on the grammar should aim at covering those error types that were found to be treatable by means of partial parsing and local error rules in the analysis of the error material. The majority of the examples are chosen form the error database and the report on the error typology. They are chosen to illustrate some of the various contexts in which each error type may occur.

5.1.1 Implemented grammar error types

ERRORS IN THE NP: GPNP

AGGREEMENT: GPNPAG

GPNPAG01 Number agreement	*Efter förberedelser av sina nya utrikesminister, Mrs Albright, som hade ett möte med sin kollega Primakov, har den rullstolsbundne Clinton träffat Jeltsin i Helsingfors. *Det slutgiltiga siffrorna får vänta. *Gäspningar och liknande beteenden skulle ha en kopplingar till aggression och sårbarhet. *Så jag får nöja mig med ett telegram och säga att de här dagen trodde jag aldrig att jag skulle få uppleva för 50 år sedan. *Det många mörka vintertimmarnas slit var tungt.
GPNPAG02	*En eventuellt segerfest får vänta.
Gender agreement	
GPNPAG08 Number agreement: noun - apposition	*Thage G Pettersson har skyllt på sina företrädare Anders Björk.
GPPNPAG03 Wrong species in the head noun	*De kanske mest personliga områden är de som nu lyfts fram.
GPNPAG14 wrong species in certain adjectives	*Barnen får använda sin egna energi.

CASE: GPNPCA:

GPNPCA01 Wrong case in a common noun	*Inför lördagen hemmapremiär var spänningen stor.
GPNPCA02 Wrong case in a proper noun	*Troligen går du inte i land med att själv hitta några ätbara svampar i vår Herres hage och inte i Pettersson hage heller för den delen.

SPECIES: GPNPSS:

GPNPSS01 Definite article missing or definite form instead of indefinite	*Kylbilen med finska, estniska och svenska flaggorna för tankarna till katastrofen. *Han går till närmaste Konsumbutiken för att köpa frukt.
GPNPSS02 Definite form after genitive attribute	*Halva regeringens mandatperioden har passerat.
GPNPSS03 Indefinite article missing in indefinite singular NP or definite article missing in definite singular NP	*Äntligen kvinnlig biskop? *Men allt överskuggande problemet för Samhall nu är den höga arbetslösheten.
GPNPSS04 Definite form before necessary relative clause	*Vi kan inte förvägra dessa länders medborgare den frihet och de ekonomiska möjligheterna som ett EU-medlemskap skulle ge.

ERRORS IN THE AP: GPAP

AGREEMENT: GPAPAG

GPAPAG01 Disagreement: parallel adjectives	*En upptrappad psykologiska krigföring väntar.
GPAPAG02 Disagreement: coordinated adjectives	*Saknade faktiskt och praktiska möjligheter att hävda sig.

ERRORS IN THE VERB SEQUENCE: GPVF

GPVFAI01 Infinite form instead of finite; subordinate clause	*Om människor börja tro på en förändring, så blir allt bättre.
GPVFAM02 Wrong verb form after modal	*Hur trygghet inte längre kan var statisk utan ligga i förnyelsen, utvecklingen och förändringen.
GPVFAM03 wrong verb form after auxiliary	*Polisen har hörde flera vittnen under kvällen och utredningen kommer att fortsätta under tisdagen.
GPVFIP01 Finite form after ”att”	*Han har lovat att i alla fall skall slå Turkiet.
GPVFMF01 Two finite verbs	*Det blev bytte dock namn i samband med den första privatiseringen under Thatcherepoken.
GPVFMF04 Infinite form instead of finite; main clause	*De avskedade kvinnorna få rådet att starta eget.
GPVFMF05 Supine instead of imperative	*Betänkt också de anläggningskostnader som tillkommer.
GPVFOP01 Double s-passive	*Saken har försökts tystas ner.

GPVFTS03 Dennis. Double supine	*Vi hade velat sett en större anslutningstakt, säger

STRUCTURAL ERRORS

WORD ORDER at CLAUSE LEVEL: GPWO

GPWOAB03 finite verb adv => adv finite verb in subordinate clauses	*Men vi måste ändå begränsa oss på grund av att det saknas framför allt tid i hallarna.
GPWOAB04 infitive adv => adv infinitive	*Man kan tro inte sina öron.
GPWOIN01 inversion => no inversion	*Jag undrar vad gör de små busungarna.
GPWOIN02 no inversion => inversion	*Nu man kan testa de kommande versionerna av programvaran.

ERRORS IN CONJUNCTIONS: GPCN

COMPOUND CONJUNCTIONS: GPCNCC

GPCNCC02 Second member missing	*Om glädjebeskedet som omvandlades till en chock som vände upp och ned på hela deras tillvaro och höll på att krossa såväl hälsa, äktenskap och ekonomi.

VERB VALENCY ERRORS: GPVV

GPVVMV01 Finite verb missing	*Man kanske inte behov av större resurser.
GPVVIP01 "att" missing after some verbs	*Vad jag förstår kommer Hälsingborgshem skicka upp 12 miljoner till skatteministern.
GPVVIP02 "att" missing after preposition	*Vidare ska pengar omfördelas till bland annat satsningar på Internet för stödja myndigheters och företags miljöarbete.
GPVVIP03 "att" doubled	*Att Sveriges ekonomi är stark igen kommer att märkas i människors vardag och det kommer att att märkas i kampen för jobben.
GPVVIP04 "att" to be removed	*Sverige började att klassa kärnkraftsincidenter enligt den internationella standarden.
GPVVPC05 Passive after some verbs	*Huset ämnar byggas.

AGREEMENT BETWEEN NP AND AP: GPAG

GPAGNA01 wrong number in the complement	*Tävlingen blev väldigt besvärliga.
GPAGNA03 wrong gender in the complement	*LO-distriktet i Stockholm är negativ och poängterar vikten av att alla elever uppnår Högskolekompetens.

PRONOUN CASE

ERRORS IN THE PP: GPPP

GPPCOF01 Wrong pronoun case	*För de som verkligen använder katalogen var det bra.

5.1.2 Word errors

SYSTEMATIC SPLIT COMPOUNDS: SEWF

SEWFSW01 Split proper noun compound	*Upplands kusten => Upplandskusten
SEWFSW13 Split proper noun compound, hyphen missing	*IT fakulteten => IT-fakulteten

5.1.3 Graphical errors

PARENTHESES: GRPA

GRPAPP Right parenthesis missing	* Nästa etapp innebar säkring av brottet, sprängning och utplanande av kalkmassorna (1994 hade 5 000 kubikmeter sprängts!

PUNCTUATION: PUES

PUESEC03 Period instead of question mark	*Är det rättvist och solidariskt.

6 Evaluation and validation strategy

Due to time limitations in the project, evaluation and validation of the Swedish SCARRIE prototype had to start before the work on the grammar was finalised according to the project plans. This had as a consequence that not all the error types that are currently implemented could be taken into account in the evaluation and validation work. Further evaluation and validation is thus called for before the prototype may be considered finalised with the error coverage presented above. According to the validation strategy that was applied, the whole grammar was tested at the same time. This is a kind of testing that has to be performed. However, prior to that, it would have been valuable to test the different error types one by one, in order to get an overview of the "competing" correct contexts that have to be covered by positive rules in order to avoid false alarms.

7 Conclusions

Work on a Swedish grammar checker based on partial parsing and the application of local error rules has shown that this is a viable approach to grammar checking directed towards formal errors. The linguistic limits of the approach were set in an earlier study (Wedbjer Rambell 1998b) with regard to a detailed error typology (Wedbjer Rambell 1998a) of more than 500 error types.

It has further been demonstrated that the chart-based ScarCheck implementation of the approach provides a well functioning and appropriate technology for the purpose. A grammar covering a subset of the error types that can be handled in this framework has been defined. Further work towards a larger span of error types should continue in accordance with the error analysis that was made in the project.

The fundamental problem when working with a grammar checker based on partial parsing is to avoid false alarms due to lexical ambiguity and false sentence segmentation. Erroneous constructions that are captured by the grammar may coincide with correct ones. The general strategy provided by ScarCheck to handle these problems is based on a grammar that generates both the correct and the incorrect analysis. If both analyses are recorded in the chart, the correct analysis "neutralises" or "covers" the erronoeus one. In order to make full use of this strategy, systematic studies of these cases should be made. The vast newspaper material that was collected in the project provides a rich basis for such studies. However, due to time limitations, such a study had to be left aside, and only those cases that occurred in the validation material or that came to mind could be considered.

References

Carlsson, Mats, 1981, *Uppsala Chart Parser 2: System Documentation* Report No. UCDL-R-81-1. Uppsala University. Center for Computational Linguistics. [revised 1982]

Sågvall Hein, Anna, 1983, *A Parser for Swedish. Status Report for Sve.Ucp. February 1983.* Report No. UCDL-R-83-2. Uppsala University. Center for Computational Linguistics.

Sågvall Hein, Anna & Starbäck, Per, 1998 *A test version of the grammar checker for Swedish.* SCARRIE. Deliverable 6.5.1. This issue.

Sågvall Hein, Anna, 1998a, *A chart-based framework for grammar checking. Initial studies.* Proceedings of Nodalida 1998. Copenhagen.

Sågvall Hein, Anna, 1998b, *A specification of the required grammar checking machinery.* SCARRIE. Deliverable 6.5.2, version final 1.1. This issue.

Starbäck, Per, 1999, *ScarCheck – a Software for Word and Grammar Checking.* This issue.

Wedbjer Rambell, Olga, 1998a, *Error Typology for Automatic Proof-reading Purposes*. SCARRIE, Deliverable 2.1, version 1.1. Uppsala University. Department of Linguistics.

Wedbjer Rambell, Olga, 1998b, *Three types of grammatical errors in Swedish.* SCARRIE, Deliverable 6.2.3, version 1.0. Uppsala University. Department of Linguistics.

Wedbjer Rambell Olga et al., 1998, *An Error Database of Swedish.* SCARRIE, Deliverable 2.1.3.2, version final 1.0. Uppsala University. Department of Linguistics.

Weijnitz, Per, 1999, *Uppsala Chart Parser Light. System Documentation.* This issue.

Appendix A:

A summary of the UCP operators

Chart building operators

`process(arg)`	inserts an active edge from and to the final vertex of the active edge; `arg` is the name of a dictionary or a grammar rule
`majorprocess(arg)`	inserts an active edge from and to the initial vertex of the active edge; `arg` is a grammar rule
`advance(arg)`	inserts an active edge from the initial vertex of the active edge to the final vertex of the inactive edge; `arg` is a subrule name; if it is left out, the next operation in the rule sequence will be executed
`store`	inserts an inactive edge from the initial vertex of the active edge to the final vertex of the inactive edge; it inherits its feature structure from the active edge
minor`store`	inserts an inactive edge from the initial vertex of the active edge to the final vertex of the active edge; it inherits its feature structure from the active edge

Operators for test and assignment

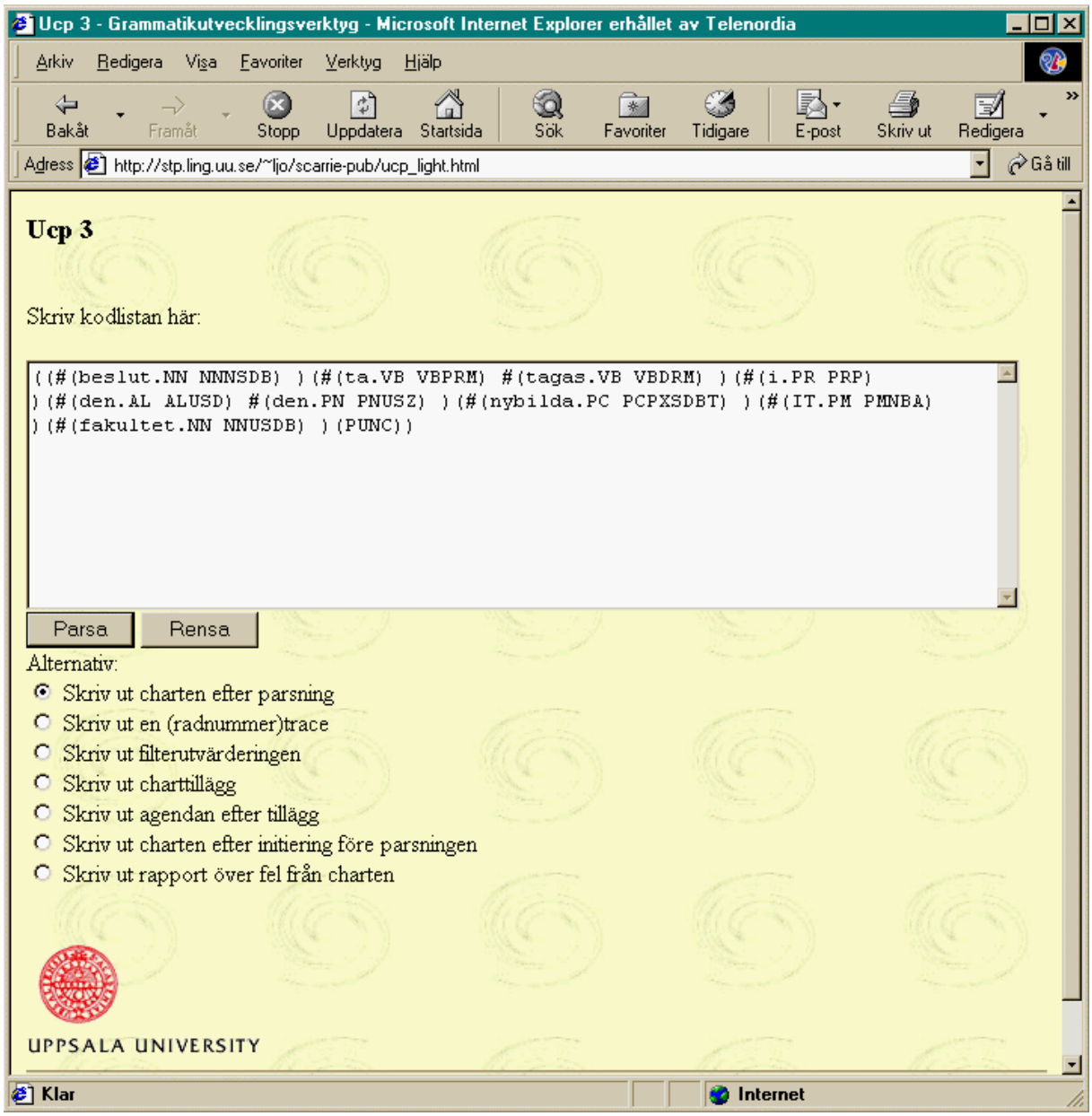
Unification	:=:	
Equality	=	
Not	`not`	
Path	`<* val1 …valn>`	denotes the value of the attribute specified by the path from `*` to `valn` in the feature structure of the inactive edge
	`<* char :property>`	denotes the value of `property` associated with the character attribute of the inactive edge in the dictionary of characters

	`<* lem :property>`	denotes the value of `property` associated with the lemma attribute of the inactive edge in the dictionary of lemmas
	`<& val1 … valn>`	denotes the value of the attribute specified by the path from `&` to `valn` in the feature structure of the active edge
	`<& val1 … valn :new>`	generates a new integer attribute in the feature structure of the active edge starting by 1 and adding 1 with every new call
	`<& val1 … valn :last>`	denotes the value of the last attribute in the path specified by `val1` to `valn` in the feature structure of the active edge
Nil	`nil`	a symbolic feature value that unifies with any other value
Atom	`'atom`	a distinct symbolic feature value

Control operators

Sequence	`(op1 , op2 , … , opn)`	Boolean 'and'
Dependent disjunction	`(op1 / op2 / … /opn)`	Boolean 'or'
Independent disjunction	`(op1 // op2 // … //opn)`	parallel processing
If-then	`(if op1 then op2 else op3)`	condition
Subrule	`rulename('val1, 'val2, … 'valn)`	subrule call (with optional parameters)
True value	`continue`	always true
False value	`failure`	always false

Appendix B: The UCP light development interface



Appendix C: Grammar coverage in terms of rule names

Main rules triggered by the dictionary (D), or the grammar: bottom-up via Majorprocess (M) or top_down via Process (P):

ADJP.COORD_ADJP		M	
ADJP_ADJ	D		
ADJP_ADVP	D	M	
ADVP	D		
ADVP_ADVP	D		
CL.EXPL	D		
CL.EXPL_PRON		M	
CL.INF_CONJ	D	M	
CL.QUEST_VB		M	P
CL.WHQUEST	D		
CL_CONJ	D		
CL.DECL_NP		M	
CL.DECL_XP		M	
CL_IMP		M	
CL_NP.VP		M	
CL_REL	D		P
CL_SPEAK	D		
CND_CN	D		
CNDP_CN	D		
NEC.REL.TAIL			P
NP.HYPCONT		M	
HYPCONT	D		
NP.DATE_DEN	D		
NP.MONTH_NN	D		
NP_ADJP		M	
NP_ADV		M	
NP_COMP	D		
NP_DET	D		
NP_NOUN	D	M	
NP.DET_POSS		M	
NP_NP	D		
NP_NP.COORD.NP		M	
NP_PMP	D		
NP_PNOUN	D	M	
NP_POSS	D	M	
NP_PRON	D		
NP_QUANT	D	M	
NP_SEL	D		
PP_PP	D		
PP_PREP	D		
QP_ADV		M	
SEARCH.RIGHT	D		
SPLITC1	D		
SPLITC2	D		
SPLITC3	D		
VP_COP.SUP		M	
VP_NP.SUP		M	
VP_PCP.FRAG	D		
VP_VERB.NP	D		
VP_VP	D		
VPF	D		
VPFC	D		
VPFE	D		
VPFO	D		
VPI	D		
VPS	D		

Follow-up rules invoked by the grammar via Advance:

```
ACOMPL
ADJP.COORD_ADJP
ADJP.COORD_CONJ
ADV.OR.VFIN.
CL.EXPL_NP
CL.EXPL.NP_VERB
CL.IMP_ADV
CL.IMP_NP
CL.NP_VFIN
CL.QUEST.VB_NP
CL.REL_ADVP.OR.NP
CL.REL_XP.
COMMA_REL.
DEN.REL.TAIL.
DET.PN.TAIL
DET.REL.TAIL
FIRST_NEXT.
FÖR.NP_SEDAN
INF_COMPL
KOMMA.VB
NEC.REL.TAIL
NOUN.TAIL
NP.ADJP_NOUN
NP.GRADE_NP
POST.ATTR_PP
NP.COORD.CONJ
NP.DET_ADJP.OR.NOUN
NP.DET.ADJP_NOUN.
NP.DET_POSS
NP.DET_QUANT
NP.DET_SEL
NP.POSS_NP
NP.PRON_PP
NP.TITLE_NP
NP_NOUN
NP_QUANT
NP_REL.CLAUSE
NP_SEL
PP.COORD
PP.COORD.CONJ
PP.PREP_NP
SEARCH.PASS
SEARCH.QUESTM
SECOND.PH.
VAD
VP.AUX.ADV_SUP.
VP.AUX_ADVP.OR.SUP.
VP.VERB_NP
VPF.VBFIN_VERB.
VPF.MOD_ADVP.OR.MAIN.
VPF.MOD.ADV_MAIN.
```

Sub-rules:

```
ADVERB
ASSIGN
ASSIGN.ERR
ASSIGN.MAJORPROCESS
ASSIGN.NOERR
```

```
ERROR
FÖR.SEDAN
NOERROR
PROPAGATE.ERR
```