

# A Chart-Based Framework for Grammar Checking Initial Studies

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

## 1 Introduction

A language checker, typically, has two basic components, a spell-checker and a grammar checker. Whereas the spell-checker, usually, limits its operation to the inspection and correction of individual text words, the grammar checker has to cope with errors that can only be detected in contexts that are larger than the word. Among the latter we find not only syntactic errors in the traditional sense, but also punctuation errors, soft lower case errors, and other violations of graphical conventions (Wedbjer Rambell 1998). Here we will only discuss errors that may be denoted construction errors, i.e. syntactic errors and certain types of punctuation errors, e.g. separators such as comma. Spelling errors that result in proper words but wrong constructions and thus cannot be captured by a spell-checker also belong to this group. For the handling of soft lower case errors and other violations of graphical conventions including erroneous use of dash, quotations mark etc. a simpler machinery may be envisaged.

We will start by examining different levels of functionality of a grammar checker. Then we will study what can be achieved by a combination of robust partial parsing and the application of local error rules. In specific, a chart-based implementation of such a framework will be presented and from our experience with this framework we will draw some conclusions.

## 2 Functionality of a grammar checker

In the first place, a grammar checker has to *detect* a construction error. Further, in order to *correct* it, or recommend a correction, it must provide a *diagnosis*. As regards the diagnosis, Uszkoreit (1996), in addition, suggests a level of *recognition*. Recognition means identifying the nature of the error in terms of localisation and constraint violations, e.g. violation of subject – verb agreement. The diagnosis should identify the source of the error as a basis for correction, e.g. changing the subject, or changing the verb. Such a four level scheme seems to provide a useful framework for the discussion of grammar checking functionality.

1. Detection
  - Identification of segments possibly containing an error
2. Recognition
  - Localisation and identification of constraints possibly violated
3. Diagnosis
  - Identification of possible error sources
4. Correction
  - a) finding or construction alternatives
  - b) ordering of alternatives
  - c) substituting most highly ranked alternatives

**Figure 1: Possible tasks in grammar correction (from Uszkoreit 1996)**

---

A parser with a complete grammar of the language or language fragment to be checked appears to be the primary tool for error detection. It will be capable of making a primary division between correct and incorrect sentences according to its grammar. All constructions outside the scope of the grammar will be flagged as erroneous. It will not, however, per se provide the functionality that is required for the recognition and diagnosis of an error. In order to do so, the parser must have quite detailed knowledge of the kinds of errors that may occur. A grammar rule may be violated in several ways, and there is no limit to the kinds of errors that may occur, at least if accidental performance errors are to be considered. Consequently, a complete account of all possible kinds of errors is out of reach. This is the motivation for the high priority given to error collection and error modelling in the Scarrie project<sup>1</sup>.

Approximately 9,000 proof-reading errors have been analysed and stored in an error database, the Scarrie *Error Corpora Database, ECD*. Each database entry comprises an error fragment and its correction marked with an error type code in accordance with an error typology that was also developed in the Scarrie project (see Wedbjer Rambell 1998). The error material was provided by two prominent Swedish newspapers, *Svenska Dagbladet*, a contracted partner of the project, and *Upsala Nya Tidning*, one of its subcontractors. See further (Wedbjer Rambell et al. 1998).

Below we will examine the applicability of the Uszkoreit scheme to the Scarrie error typology. The examples will all be chosen from the ECD.

A first example from the ECD:

---

Error text	Proof-reader's correction	Error type code
<i>*det tidiga 1800-talen</i>	<i>det tidiga 1800-talet</i>	GPNPAG01

---

<sup>1</sup> The Scarrie project aims at the development of a proof-reading tool for the Scandinavian publishing industry, see further url:<http://guagua.echo.lu/langeng/en/le3/scarrie/scarrie.html>.

The error type code reflects the four levels of the typology.

Group	GP: grammar problem
Category	NP: nominal phrase
Subcategory	AG: agreement
Specification	01: number

The proof-reader has chosen to correct the noun, changing its number from plural to singular. However, a correction of the determiner by a change of number from singular to plural is also an alternative to consider. The example with the two possible corrections fits well into the Uszkoreit scheme:

---

*\*det tidiga 1800-talen*

Detection:	NP is flagged
Recognition:	violation of number agreement in premodifier – noun
Diagnosis:	1. plural instead of singular in noun 2. singular instead of plural in premodifier
Correction:	1. <i>det tidiga 1800-talet</i> (one correction step) 2. <i>de tidiga 1800-talen</i> (one correction step)

**Figure 2: A simple example from the ECD in the Uszkoreit (1996) framework**

---

Below we present an example of an error that can be analysed in more than one way. The first analysis is in accordance with the correction made by the proof-reader. The second one was found by ScarCheck, our prototype grammar checker (to be described below).

---

*\*det slutgiltiga siffrorna*

Detection:	NP is flagged
Recognition:	1. violation of number agreement in premodifier - noun 2. violation of gender and number agreement in premodifier - noun
Diagnosis:	1. singular instead of plural in premodifier 2. neuter instead of utrum in premodifier and plural instead of singular in noun
Correction:	1. <i>de slutgiltiga siffrorna</i> (one correction step) 2. <i>den slutgiltiga siffran</i> (two correction steps)

**Figure 3: An example of alternative error analyses in the Uszkoreit (1996) framework**

---

The accommodation of this example in the Uszkoreit scheme is not quite straightforward. Splitting the recognition level up into two, i.e. the localisation of the error (premodifier - noun agreement), and the identification of the constraints possibly violated is an alternative to be considered. Apart from that, we think that this scheme provides a good basis for discussing and comparing grammar checking functionality.

In its present version, the ScarCheck grammar checking prototype does not go beyond the recognition level. It does, however, provide mechanisms for ordering alternatives at this level.

### 3 Constraint relaxation and the application of local error rules

Bustamante&Sánchez (1996) make a primary division of grammatical errors into non-structural and structural errors. Non-structural errors are mismatches of features that do not affect configurational issues. Examples of non-structural errors are agreement violations. Structural errors are mismatches of features that represent syntactic categories. Examples of structural errors are wrong head-argument relations, word order errors, and substitution of categories. For the handling of non-structural errors the authors propose constraint-relaxation techniques. Such techniques are also used for the detection of structural violations concerning the use of prepositions in head-argument relations. The approach has been implemented in prolog, and applied to Spanish. The resulting demonstrator covers intra- and intersyntactic agreement, errors due to wrong usage of bound prepositions in head argument relations, and errors on portmanteau words. Certain stylistic aspects are also covered. The authors also provide a comprehensive background to grammar checking.

A similar approach has been chosen for Swedish. The main difference is to be found in the Swedish attempt to use an underspecified grammar in order to achieve a realistic coverage. Thus a combination of robust partial parsing and the application of local error rules is being explored. Phrase constituents (NP, AdjP, AdvP, PP, VP (verbal core)) are analysed by means of rules that accept feature violations. Anticipated errors at clause and sentence level, be they structural or non-structural, are handled by means of local error rules. The local error rules operate on the results of the parsing process. They may be formulated in terms of phrase categories, lexical categories, lemmas, and morpho-syntactic features. All the rules are integrated in one single parsing grammar. Typically, each rule covers both the positive and the negative case.

#### 3.1 ScarCheck - a chart-based implementation of a grammar checker

The grammar checking strategy outlined above has been implemented in a chart-based framework. The prototype checker, *ScarCheck*, has two basic modules, a chart parser and a chart scanner. The parser builds as much structure as the grammar allows, and the scanner traverses the chart collecting and reporting errors. Below we present an example of the interaction with ScarCheck. (So far, there is no interface connecting the two modules). For the sake of the presentation, comments in quotation marks are added.

```

"User: invoking the parser:"
  > (p "de tidiga 1800-talet ")           'the early 19th century'

"System: respons"
  1 parse, 22 vertices.
  T

"User: printing the result:"
  >(pr)

"System: respons"
  de tidiga 1800-talet :
  (* = (PHR.CAT = NP
        FIRST = +
        DEF = DEF
        FORM = DEF
        NUMB = PLUR
        GENDER = NIL
        DET = (LEM = DEN1.AL
              WORD.CAT = ART)
        ATTR = (PHR.CAT = ADJP
              HEAD = (1 = (LEM = TIDIG.AV
                          WORD.CAT = ADJ
                          DEGREE = POS))
              GENDER = NEUTR
              NUMB = < * NUMB >
              DEF = NIL2
              FUNC = ATTR
              A-FORM = A
              CASE = BASIC
              SEX = NIL)
        ERR = (1 = GPNPAG01)
        CASE = BASIC
        HEAD = (WORD.CAT = NOUN
              LEM = 1800-TAL.NN)))

  OK

"User: invoking the chart scanner"
  > (reportchart)

"System: respons"
  FELTEXT 'erroneous text':   de tidiga 1800-talet
  FEL 'error':               number agreement in premodifier - noun

```

**Figure 4: An example of a simple interaction with ScarCheck**

---

The error message generated by `reportchart` is an interpretation of the (first and only) error feature value (`ERR = (1 = GPNPAG01)`) that was found in the chart. The message provides two kinds of information, i.e. the local context in which the error was recognised, and a spelling-out of the error type code. In the example the local context is identical to the full input. This is not always so, as we will see below. First, however, we will present a case where two errors are found:

---

<sup>2</sup> The NIL value unifies with any value.

```

>(p "Det slutgiltiga siffrorna ")          'the final figures'

1 parse, 27 vertices.
T
> (reportchart)

FELTEXT: Det slutgiltiga siffrorna
FEL:      1 number agreement in premodifier - noun
          2 gender agreement in premodifier - noun

```

**Figure 5: Two errors recognised by ScarCheck**

---

The two errors are presented in the same order as they appear in the chart. This is also the preferred order. Violation of number agreement is, by far, the most common agreement violation in the ECD. Ordering the errors according to priority is the task of the grammar rule writer.

### 3.1.1 Parsing for errors

The parser, UCP (Sågvalld Hein 1983) 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, but a sequence of edges representing words and phrases. The grammar is formulated in a procedural formalism (Sågvalld Hein 1983), and grammar rules (incl. local rules of anticipated errors) are triggered from the grammar. For instance, NP rules are triggered at the recognition of lexical categories that may appear as NP introducers.

Clause initial position and clause final position are important clues to the detection of structural errors at sentence level. As a means of identifying the beginning of a clause when it appears as the first constituent of the sentence, chart initial position is recorded as a feature in the morphological description of the first word of the input. An example of a rule where such a signal is motivated is the local error rule designed to catch erroneously deleted finite verbs in main clauses introduced by NPs. The invocation of the rule is conditioned by the position of the NP; it is only invoked if the NP appears at the potential beginning of a clause.

Unification of feature structures is the basic operation for test and assignment in the procedural UCP formalism.

Errors are recorded as features with values set in accordance with the error typology. In principle, the parser accepts any number of errors in a constituent.

Reportchart expects the errors to be located at the top level of the feature description. Consequently, the parser has to account for proper error propagation.

#### 3.1.1.1 Error propagation.

For instance, if an error is found in one of two coordinated NPs, it has to be propagated to the top level of the description of the coordinated NP (see fig. 6). It may be analysed in several ways. In particular, there are several options as regards the scope of the determiner (*det*) and the adjective (*större* 'bigger'). Do they modify both nouns (*maskinerna* 'machines' and *traktorerna* 'tractors') or only the first one? In cases like this one, the parser will provide only one analysis, and it will choose the most superficial one according to which the premodifiers

modify only the first, and closest, noun. This decision is motivated by processing economy; we want the checker to be as fast and efficient as possible.

Let's examine the consequences of this choice. Two (alternative) agreement errors will be recognised in the first NP, whereas there are no errors to be found in the second one consisting of a singular noun. The errors that are found in the first NP are propagated to the top level of the description, i.e. the level of the coordinated NP. The coordinated NP constitutes the error context and will be presented as such to the user (see fig. 7). It may be argued that the context is too wide and thus counterintuitive. If so, the most obvious solution would be to make Reportchart take a deeper look into the structure at the expense of processing economy. We would not, however, be in a better position in this respect by making a "deeper" analysis.

det större maskinerna och traktorerna 'the bigger machines and tractors':

```
(* = (PHR.CAT = NP
  ERR = (1 = (1 = GPNPAG01
            2 = GPNPAG02))
  1 = (PHR.CAT = NP
      FIRST = +
      DEF = DEF
      FORM = DEF
      NUMB = SING
      GENDER = NEUTR
      DET = (LEM = DEN1.AL
            WORD.CAT = ART)
      ATTR = (PHR.CAT = ADJP
            HEAD = (1 = (LEM = STOR1.AV
                          WORD.CAT = ADJ
                          DEGREE = COMP))
            GENDER = <* 1 GENDER>
            NUMB = <* 1 NUMB>
            DEF = NIL
            FUNC = ATTR
            A-FORM = NIL
            CASE = BASIC
            SEX = NIL)
      ERR = <* ERR 1>
      CASE = BASIC
      HEAD = (WORD.CAT = NOUN
            LEM = MASKIN.NN))
  2 = (LEM = OCH.CN
      WORD.CAT = CONJ)
  3 = (PHR.CAT = NP
      NUMB = PLUR
      GENDER = UTR
      CASE = BASIC
      DEF = DEF
      HEAD.FORM = <* 3 DEF>
      HEAD = (WORD.CAT = NOUN
            LEM = TRAKTOR.NN))))
```

**Figure 6: An example of a propagated error**

---

```
> (reportchart)
```

```
FELTEXT: det större maskinerna och traktorerna
FEL:      1 number agreement in premodifier - noun
          2 gender agreement in premodifier - noun
```

**Figure 7: A report of a propagated error**

---

### 3.1.1.2 Changing or postponing a decision

Sometimes the parser needs to change its decision about an error. For instance, in a Swedish NP introduced by the definite determiner *den*, the head noun should be in the definite form, e.g. *de områdena* 'those areas' unless it is followed by a restrictive relative clause introduced by the pronoun *som* 'which'. In the latter case, the indefinite form is the proper one, e.g. *de områden som* 'those areas which'. The parser can handle such problems using its look-ahead facility (see further Sågvall Hein 1983, p. 12) in combination with an error cancelling NOERR feature feature. It neutralises an ERR feature with the same value. For instance, *de områden* will be analysed as an NP with an agreement violation of the species value: GPNPAG03 (see fig. 8) whereas the error will be cancelled in the analysis of *de områden som* (see fig. 9).

```
de områden :
(* = (PHR.CAT = NP
      FIRST = +
      DEF = DEF
      FORM = DEF
      NUMB = PLUR
      GENDER = NEUTR
      DET = (LEM = DEN1.AL
             WORD.CAT = ART)
      ERR = (1 = GPNPAG03)
      HEAD = (WORD.CAT = NOUN
              LEM = OMRÅDE.NN)))
```

```
FELTEXT: de områden
FEL:      1 species agreement in premodifier - noun
```

**Figure 8: An error that may be cancelled**

---

```
de områden [som] :
(* = (PHR.CAT = NP
      FIRST = +
      DEF = DEF
      FORM = DEF
      NUMB = PLUR
      GENDER = NEUTR
      DET = (LEM = DEN1.AL
             WORD.CAT = ART)
      ERR = (1 = GPNPAG03)
      HEAD = (WORD.CAT = NOUN
              LEM = OMRÅDE.NN)
      NOERR = (0 = GPNPAG03)))
```

**Figure 9: Cancelling an error**

Reportchart will consider the ERR feature cancelled by the NOERR feature. It will face a situation where there are two competing edges of the same length, one with an error in it and one with no error (error cancelled). It will prefer the error free one.

### 3.1.1.3 Parsing local error rules

Whereas a robust grammar rule recognises configurationally well-formed units, a local error rule, typically, applies to fragments of constituents. Such fragments are represented by edges in the chart but not used in the further analysis. For instance, a local error rule will apply to the initial fragment of the main clause *Det bli sång och musik* 'There will be song and music' and recognise an error in the verb form (infinitive/imperative instead of finite), see fig. 10. The rule was designed to capture non-structural violations of the verb-second rule in Swedish main clauses.

```
Det bli [sång och musik] :                'There will be song and music'
(* = (ERR = (1 = GPVFFV01)
      PHR.CAT = CL.FRAG))

FELTEXT: Det bli
FEL:     1 infinite verb instead of finite
```

**Figure 10: A non-structural error recognised by a local error rule: main clause**

---

An analogous rule was formulated for the recognition of improper verb forms in the position of the finite verb in subjunctive clauses, see fig. 11.

```
Om människor börja [tro] :                'If people begin [to believe]'
(* = (ERR = (1 = GPVFFV01)
      PHR.CAT = CL.FRAG))

FELTEXT: Om människor börja
FEL:     1 infinite verb instead of finite
```

**Figure 11: A non-structural error recognised by a local error rule: conditional clause**

---

In the ECD there are also examples of deleted finite verbs. Since the position of the finite verb is fixed in Swedish, deleted finite verbs may be recognised by means of local error rules, provided that they are capable of recognising the preceding constituent (and, in subjunctive clauses, an optional adverb) in its full extension. Hereby, heavy demands are made on the parser's coverage, in specific, with respect to NPs. An example of a deleted finite verb that is recognised by means of a local error is presented in fig. 12. In this case the parser easily found the preceding constituent.

```

Det nödvändigt [att tänka i nya banor] : 'It necessary [to think in new
                                         ways]
(* = (PHR.CAT = CL.FRAG
      ERR = (1 = GPVVMV01)))

FELTEXT: Det nödvändigt
FEL:      1 finite verb missing

```

**Figure 12: A structural error recognised by a local error rule**

---

### 3.1.1.4 Partial parsing

The parser starts its operation at the level of the characters. It records unknown strings, and saves them for the protocol. When a rule fails or there is no applicable rule it just moves on to the next word and the rules that are associated with that word are triggered. For instance, in fig. 13 we present the error messages that were generated as a result of the analysis of a sentence in three independent segments. There is no edge covering the whole sentence (0 parses) but three partial parses are stored in the chart and interpreted by the chart scanner. The first segment was analysed by means of a local error rule, the second was reported missing from the dictionary, and the third segment was analysed by means of a robust NP rule.

The example *Det bli förmodligen det slutgiltiga siffrorna*. 'It become probably the final figures'. was constructed for the sake of the presentation, and for the same reason the adverb *förmodligen* was removed from the dictionary. (Sentences with this simple structure are likely to be covered fully by the grammar.)

```

> (p "Det bli förmodligen det slutgiltiga siffrorna. ")

0 parses, 48 vertices.
NIL
> (reportchart)

FELTEXT: Det bli
FEL:      infinite verb instead of finite

FELTEXT: förmodligen 'probably'
FEL:      Ordet finns ej i lexikonet. 'The word is not in the dictionary.'

FELTEXT: det slutgiltiga siffrorna
FEL:      1. number agreement in premodifier - noun
          2. gender agreement in premodifier - noun

```

**Figure 13: Error messages based on partial parsing**

---

It is of vital importance that the parser is capable of recovering when rules fail or there are gaps in the grammar or the dictionary. Otherwise, it will not be capable of recognising errors that appear after that initial part of the sentence that is covered by grammar rules<sup>3</sup>.

As is normally the case in chart parsing, the parser ends its work when no more rules apply and the agenda is exhausted.

---

<sup>3</sup> An alternative framework for grammar checking in which robust parsing may be combined with the application of rules of anticipated errors has been proposed and implemented by Vosse (1994). The parser, basically, works top down and there seems to be no way of implementing partial parsing in an efficient way.

### 3.1.2 Scanning the Chart for Errors

The chart scanning module Reportchart traverses the chart in search for error features. Starting at the first vertex, it inspects the top level of the feature description of the longest inactive edge. If an error is found, a corresponding message is formulated and taken to the error protocol, otherwise the inspection just goes on to the final vertex of the current edge and the edgeset going out from that vertex. When there is a choice between several edges, the scanner always chooses the longest one disregarding the others. When there are more than one edge of maximum length, and one of them has no errors, this edge is preferred and no error is recorded in the protocol. Before finally accepting an error, Reportchart verifies that the error has not been cancelled by means of a corresponding NOERR feature.

## 4 Conclusions and future work

The proposed approach has been tested in the implemented framework, both with regard to feature relaxation techniques for the handling of non-structural errors and the application of local error rules for the handling of structural errors and some types of non-structural errors at clause level. The results that were achieved so far are encouraging.

In specific, the handling of non-structural errors is found to be fairly straightforward. An issue to be further investigated though is the number of feature violations to be accepted for the various phrase types. If too many errors are accepted, the checker may overgenerate. Another problem that should be further explored concerns feature propagation. When and how far should an error feature be propagated?

The coverage of the checker will be modelled after the Scarrie Error Corpora Database. Preliminary studies indicate that a great number of error rules will be needed to account for the variety of error types that are represented in the base.

The prototype checker is also being evaluated in an application to controlled language at Scania (Sågvall Hein et al. 1997).

## References

- Bustamante, Flora Ramírez & León, Fernando Sánchez, 1996, *GramCheck: A Grammar and Style Checker*, in Proceedings of the 16<sup>th</sup> International Conference of Computational Linguistics (Coling -96): 175 – 181.
- Sågvall Hein, Anna, 1983. *A Parser for Swedish. Status Report for Sve.Ucp. February 1983*. Report No. UC DL-R-83-2. Uppsala University. Center for Computational Linguistics.
- Sågvall Hein, Anna, Almqvist, Ingrid, & Starbäck, Per, 1997. *Scania Swedish – A Basis for Multilingual Machine Translation*. Translating and the Computer 19. Papers of the Aslib conference held on 13 & 14 November 1997.
- Starbäck, Per, forthcoming, *ScarCheck – a Software for Word and Grammar Checking*. Technical Report. Uppsala University. Department of Linguistics.

Uszkoreit, Hans, 1996. *Grammar Checking. Theory, Practice, and Lessons learned in LATESLAV*. Concluding oral presentation at the final review meeting of the Lateslav Project (PECO 2824). Prague. August 1996.

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.

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

Vosse, Theo G., 1994. *The Word Connection. Grammar-based Spelling Error Correction in Dutch*. Amsterdam.