UPPSALA UNIVERSITY

Project ref. no. *LE3-4239*

Project title *SCARRIE Scandinavian Proof-reading Tools*

Deliverable number *DEL 6.5.2*

Deliverable title *A specification of the required grammar checking machinery*

Number of pages *39*

WP/Task responsible *Anna Sågvall Hein, Department of Linguistics, Uppsala University, Box 513, S-751 20 Uppsala, Sweden*
Email: anna@ling.uu.se

Author(s) *Anna Sågvall Hein*

EC Project Officer *Ray Hudson, Antonio Sanfilippo*

Keywords *Grammar checking, specification, robustness, unrestricted text, error types, error correction, performance*

Abstract *The aim of this task is to specify a grammar checking machinery for Scarrie and to identify an appropriate technology. Three investigations were carried out, i.e. explorative work on the CORRie fragment analysis approach for Danish, and Swedish, and on ScarCheck, the chart-based test version of a grammar checker for Swedish. In addition, an inquiry on commercial software was made. It was concluded that CORRie with two options for grammar checking, i.e. CORRie fragment analysis approach (for Danish and Norwegian) and ScarCheck (for Swedish) will serve as the best technology base for the Scarrie pilot.*

Executive Summary

The main objective of this task is to specify the demands on the functionality of the Scarrie grammar checking machinery and to decide on a technology baseline in accordance with this specification.

Even though the focus of this task is on grammar checking, its integration with the operation of the spell checker is vital and has to be given due attention. Accordingly, three alternatives for a technology baseline for combined spell checking and grammar checking were identified and evaluated:

1. CORRie for word checking and CORRie fragment analysis for grammar checking
2. CORRie for word checking and ScarCheck for grammar checking
3. External commercial software

Three different investigations were carried out, i.e. explorative work on the CORRie fragment analysis approach for Danish (see Paggio, P. 1998) and Swedish (see Wedbjer Rambell, O. 1998), and on ScarCheck, the chart-based test version of a grammar checker for Swedish (see Sågvall Hein, A. 1998a, 1998b). In addition, an inquiry on commercial software on the market was made.

For the identification of commercial software for combined spell checking and grammar checking a questionnaire was compiled. It also serves as a software specification and as a basis for comparing and evaluating commercial software with the two CORRie alternatives.

The results of the inquiry were merged into the questionnaire together with the results of the investigations of CORRie fragment analysis and ScarCheck (see Sågvall Hein et al. 1998). From these data it was concluded that CORRie with two options for grammar checking, i.e. CORRie fragment analysis (for Danish and Norwegian), and ScarCheck (for Swedish) would serve as the best technology baseline for the Scarrie pilot.

A specification of the required grammar checking machinery

Anna Sågvall Hein

Uppsala university, Department of Linguistics

16 June 1998

The main objective of this task is to specify the demands on the functionality of the Scarrie grammar checking machinery and to decide on a technology baseline in accordance with this specification. In pursuing this goal three different investigations were carried out, i.e. explorative work on the CORRie fragment analysis approach for Danish (see Paggio, P. 1998) and Swedish (see Wedbjer Rambell, O. 1998), and on ScarCheck, a chart-based test version of a grammar checker for Swedish (see Sågvall Hein, A. 1998a, 1998b).

In addition, an inquiry on commercial software on the market was made. The results of the inquiry were merged into a common questionnaire together with the results of the investigations of CORRie fragment analysis and ScarCheck. From these data it was concluded that CORRie with two options for grammar checking, i.e. CORRie fragment analysis (for Danish and Norwegian), and ScarCheck (for Swedish) would serve as the best technology for the Scarrie pilot.

Introduction

Even though the focus of this task is on grammar checking, its integration with the operation of the spell checker is vital and has to be given due attention. Accordingly, three alternatives for a technology baseline for combined spell checking and grammar checking were identified and evaluated:

1. CORRie for word checking and CORRie fragment analysis for grammar checking
2. CORRie for word checking and ScarCheck for grammar checking
3. External commercial software

Thus three different investigations were carried out, i.e. explorative work on the CORRie fragment analysis approach for Danish (see Paggio, P. 1998) and Swedish (see Wedbjer Rambell, O. 1998), and on ScarCheck, a chart-based test version of a grammar checker for Swedish (see Sågvall Hein, A. 1998a, 1998b). In addition, an inquiry on commercial software on the market was made.

For the identification of commercial software for combined spell checking and grammar checking a questionnaire was compiled. It also serves as a software specification and as a basis for comparing and evaluating commercial software with the two CORRie alternatives.

The results of the inquiry were merged into a common questionnaire together with the results of the investigations of CORRie fragment analysis and ScarCheck (see App. 2). It captures and summarises the data on which the conclusions regarding technology baseline are drawn.

Software specification in terms of a questionnaire

The questionnaire was produced in two versions, a general version with some basic questions for a first run (see App. 1), and a more detailed version with follow up questions (see App.2). It comprises questions on word checking as well as on grammar checking. Below we will focus on those aspects of the software specification that are vital in grammar checking and in the integration of spell checking and grammar checking. Before that, however, we will report on the first run of the inquiry of commercial software.

An inquiry on software for combined spell checking and grammar checking, 1st run

A list of companies to approach was compiled jointly by the partners and responsibilities were distributed:

Target companies and responsible partners:

Good Language Software	WF
Inso	WF
Inxight	CST
Le Correcteur	WF
LingSoft	UU
Microsoft (Word 7/8)	WF
MorphoLogic	UU
RabbitSoft	WF
Skribent	SvD
Soft-Art	WF
Tansa	SvD
Terracom	WF

Results of the first run

Company	**Partner**	**Response**
Good Language Software	WF	1
Inso	WF	1
Inxight	CST	1
Machina Sapiens (Le Correcteur)	WF	2
LingSoft	UU	1
Microsoft (Word 7/8)	WF	0
MorphoLogic	UU	2

RabbitSoft	WF	0
Skribent	SvD	1
Soft-Art	WF	1
Tansa	SvD	0

Response:

0:	no answer,
1:	NO to one or more of the first four fundamental questions
2:	YES to the first four fundamental questions

Terracom could not be identified as a language engineering company and was left out.

Machina Sapiens gave YES as an answer to the first four questions; however, they seem to have a misunderstood question No 2, focusing on robustness. The answer to this question is YES, even though the information given on the homepage of the company clearly states that the grammar checker basis its operation on complete parsing. (This is in accordance with the conclusions made by UU in the work on WP 6.1, *A study of three commercial grammar checkers*, Le Correcteur from Machina Sapiens being one of the grammar checkers that were examined.)

LingSoft and Inxight are both well-known providers of finite-state technology, a technology that has been suggested several times in the course of the project. Both companies also agree that finite state technology should be interesting as a basis for spell checking and grammar checking. However, none of the companies may provide a commercial software today with capacities for grammar checking.

It was concluded from the results of the first run that only the Hungarian company MorphoLogic was a relevant target for follow-up questions. (The full documentation of the exploration of the first run is available at UU.)

An inquiry on software for combined spell checking and grammar checking, 2nd run

2nd run comprises only three alternatives, i.e. MorphoLogic and the two CORRie alternatives. In other words, there are two alternatives for a spell checking software, i.e. MorphoLogic and CORRie, and three alternatives for a grammar checking software. A full account of the results of the investigation is presented in App. 2.

In evaluating the two spell checking alternatives we will not make a complete comparison of the answers to the questionnaire here; we only bring up some fundamental aspects of MorphoLogic that we find make it unsuitable as a spellchecker for Scarrie, i.e.

1. Vague data on resources required for conversion to another language (Is there a version for another language?)
2. Test version for Hungarian only (How can we make a test, not knowing Hungarian?)
3. Stem and affix dictionary (As motivated in TA, Scarrie will be based on form dictionaries.)

4. Vague answer to questions concerning correction principles
5. No performance figures

The main goal of this task is to specify software for grammar checking, and here we will concentrate on those aspects of the investigation that we find fundamental in grammar checking and in the integration of grammar checking and spell checking.

Robustness

A grammar checker for unrestricted text must be able to cope with incomplete grammatical data; complete parsing is no viable alternative. Two alternatives for ensuring parsing robustness were identified, fragment analysis and partial parsing. (The answers given by MorphoLogic to this issue were vague.)

Fragment analysis and shallow parsing in the CORRie framework

The CORRie parser was originally designed for complete parsing (Vosse 94). This means that for each input the parser has to build some structure spanning it from beginning to end. However, as suggested by Vosse (email communication, February 1998), a sentence may be analysed in terms of fragments that are not fully specified, rather than in traditional constituents. A shallow parse may thereby be generated.

"Although a full sentence parse must be produced, rules may be written covering a sentence in fragments. Hereby it is possible to focus on the internal structure of certain syntactic elements leaving other elements unanalysed or unidentified." (App. 2: 2.1) This approach has been explored for Danish (Paggio 1998) and Swedish (Wedbjer Rambell 1998). Errors in NPs have been in focus of both investigations.

Error recognition in CORRie is carried out by means of feature overriding mechanism built-in in the system, and by means of the application of error rules, i.e. rewrite rules rules explicitly describing incorrect patterns. Both strategies were successfully explored within the CORRie fragment approach.Being robust, the system does not crash when a parse is not produced; an error may be overlooked but that's all there is to it.

The conclusions of the two investigations of the CORRie fragment analysis approach differ slightly between the two languages:

Conclusions regarding CORRie fragment analysis for Danish

"To conclude, these experiments show that although a complete parse spanning over the whole sentence must be generated for CORRie to be able to recognise and correct an error, this parse need not be too complex or computationally expensive." (Paggio 1998).

The Danish experiment with CORRie was directed towards NPs only.

Conclusions regarding CORRie for Swedish

"It is quite possible to capture agreement errors in NPs of different syntactic complexity using the fragment analysis approach. The minor test presented in this report shows acceptable results. However, many agreement errors may not be recognised due to lexical ambiguity. [...]

To expand the grammar to embrace erroneous verb sequences and problems at clause level such as missing main verbs in the fragment analysis framework would be much more difficult to achieve compared to agreement errors in noun phrases." (Wedbjer Rambell 1998)

Partial parsing by means of ScarCheck

In the ScarCheck model robustness is ensured via partial parsing and the application of local error rules. By partial parsing we understand an approach where there need not be an analysis spanning the entire input. Only certain types of constituents are analysed, such as NPs, PPs, APs, AdvPs, and VGs (the verbal core of the VP). The constituent analysis is also robust in itself in that it allows for feature relaxation for catching feature violations, such as agreement errors. Typically, there are no sentence rules. Segments that are not covered by grammar rules are stepped by in the analysis. As opposed to the fragmens in the CORRie fragment analysis, unanalysed segments need not be foreseen in the grammar.

The rules are formulated in a procedural formalism and invoked bottom-up at the recognition of lexical categories. For instance, the recognition of a determiner leads to the invocation of an NP-rule (designed for the recognition of NPs introduced by determiners). Local error rules are formulated in the same formalism as the grammar rules and invoked in the same manner. Whereas the partial parsing rules generate linguistic descriptions that may be used by other rules in the analysis, the application of the local error rules generates descriptions of erroneous fragments of that are not to be used by other rules.

ScarCheck has only been applied to Swedish. It handles errors in NPs, APs, PPs, VGs, and at clause and sentence level (App. 2: 11.3 - 11.6) .

Error coverage

Error coverage is an important aspect when it comes to deciding on a grammar checking technology.

The Scarrie pilot is not aiming at handling all grammatical error that may occur. Only some types will be covered, i.e. a subset of those that were identified in WP 2 (see DEL 2.1.1.2, DEL 2.1.2.2, DEL 2.1.3.2). The filtering process will take error type and error recognition feasibility into account (see DEL 6.2.x), in addition to frequency and user requirements (*User requirements - Language and Typography,* Scarrie common workspace, Scarrie Users).

For an instance, according to the Swedish error data base, errors in the NP dominate (40%), followed by verb valency errors (17%), errors in the PP (11%), and errors in the VG (8%), see further DEL 2.1.3.2, p. 19). The handling of valency errors is outside the scope of the Scarrie project, the main reason being insufficient resources in terms of manpower. (Defining and including valency frames into the dictionary to the extent that would be required for handling such error types in a general fashion would need a project of its own.) User requirements include errors in the NP, errors in the VG, and several types of errors at clause and sentence levels. Consequently, the Scarrie grammar checker for Swedish aims at covering errors in the

NP, errors in the AP, errors in the AdvP, errors in the VG, some errors in the PP, and several error types at clause and sentence level.

As regards Danish and Norwegian, apart from NP agreement errors, the set of error types to be covered has so far not been finalised.

MorphoLogic reports no handling of agreement errors ("Agreement is not a critical problem in Hungarian", App 2: 11.1). It is difficult to see how agreement errors might be captured in the MorphoLogic framework, because no concrete answers to the questions concerning grammar formalism were given ("local grammar rules", App.: 11) and no illustrative example of the formalism was presented.

Error correction

The CORRie grammar checker generates corrections for those errors that are recognised as feature violation errors. It looks up the incorrect word in the dictionary, finds its lemma, and searches for an alternative word form with the correct set of features. If there are several candidates, the program chooses the alternative with the shortest edit distance to the erroneous word. No correction is generated for errors recognised by means of local error recognition rules or if the correct word form is missing in the dictionary.

So far, ScarCheck comprises no error correction mechanism. An implementation of the same principles for error correction as those used in CORRie presents no general problems.

Cooperation with spell checker

For efficiency, space, and maintenance reasons, it is important that a combined program for spell checking and grammar checking uses the same dictionary for both functions, and that dictionary search is carried out only once (see Sågvall Hein, A. 1997).

This is the case in CORRie. The main dictionary contains explicit information required for spell checking, and information required for the syntactic processing in terms of syntactic codes. Before this information may be used by the parser the codes have to be translated into the linguistic formalism (feature structure) used by the parser.

Integrating ScarCheck into the CORRie framework

ScarCheck is a viable alternative to the CORRie fragment analysis only insofar as it can be integrated into the CORRie framework, or with another powerful spell checker. In comparison with other spell checking software on the market, CORRie stands out as a rich and flexible alternative (see App. 2). Because of this, and because of the good results achieved with ScarCheck with respect to the Swedish target error types, work was initiated on integrating ScarCheck into the CORRie system, in spite of the difficulties that had been foreseen (Music, B. 1997).

Interfacing spell checking and syntactic parsing in CORRie, basically, amounts to forwarding and translating the syntactic codes that are associated with the words as they are recognised by the spell checker. If a new parser is inserted these codes have to be translated to the format used by that parser. Most words are recognised as a result of successful search in the dictionary (main dictionary of one word units, and multiword dictionary of phrasal words), and the syntactic codes may be retrieved from there. However, for words outside the dictionary, that are recognised by means of rules (e.g. compounds, proper nouns, numerical expressions) or by other means (signs of punctuation) syntactic codes have to be generated accordingly.

In integrating the ScarCheck grammar checker into the CORRie platform several technical problems had to be solved concerning the proper generation and forwarding of codes representing syntactic ambiguities, and of codes for words outside the main dictionary. As regards the concrete steps that were taken in the integration process, see App. III. Solving these problems was just as important for a successful realisation of the CORRie fragment analysis approach. Integrating a new parser into CORRie is from now on a straightforward operation that may be realised via the exchange of the translation table.

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 (Sågvall Hein 1998a). Below we present an example of the input to the ScarCheck chart parser forwarded from the CORRie spell checker.

- *Folk väntade förmodligen på det större maskinerna och traktorerna* [People were probably waiting for the bigger machines and tractors]

```
(sp '((NNNXIB ) (VBARM PCPXSDB ) (ABX ) (PR ABX ) (PNNSZ
NNNSIB ALNSD ) (AVXXXBC ) (NNUPDB ) (CN ) (NNUPDB ) ))
```

The chart parser is invoked by means of a function call "`sp`" and a quoted list of arguments. Each argument is a list of one or more syntactic codes, e.g. one code as in (`NNUPDB`) = noun, utrum, plural, definite, basic case, for *maskinerna, traktorerna*, or two alternative codes as in (`PR ABX )=` preposition or adverb for *på*. It builds an initial chart in which each syntactic code is represented by an edge of its own, and processing starts.

Reportchart scans the chart generated by the parser, and in this example it will find an edge with an error message (GPNPAG01) spanning a sequence of edges from vertex 5 to vertex 7, and generate an error message accordingly:

> (reportchart)

INTERVALL [INTERVAL]: 5,7

FEL [ERROR]: number agreement in premodifier - noun

In App. IV a few more examples of input to ScarCeck and its results are presented.

Performance

Recall and missing errors

Recall may be tested systematically only when the set of target error types has been determined. This is, basically, the case for Swedish (see above), whereas some decisions remain to be made for Danish and Norwegian[1].

Tests that were carried out so far for Swedish show that recall with respect to the target error types is satisfactory in the current implementation of ScarCheck, but not in the current implementation of the CORRie fragment analysis approach.

Tests carried out so far on Danish in the CORRie fragment analysis model show that recall with respect to fundamental NP agreement errors is satisfactory.

Precision and false alarms
The tests that were carried out so far show that satisfactory precision may be achieved in both frameworks.

Speed
Processing time depends crucially on the speed of the processor of the computer. Therefore it is not quite relevant to present figures on processing time in isolation. However, at UU a comparison was made between ScarCheck, and CORRie fragment analysis. The same test sentences were run on the same computer. Processing time for ScarCheck was roughly 3.25 times slower than for CORRie (App.2: p. 16). A factor that was not taken into account though was recall. ScarCheck detected more error types than CORRie. Still we may safely conclude that ScarCheck in its current implementation is slower than CORRie. This is not surprising. The ScarCheck parser is written in Lisp, and it comprises a machinery with many functions that are not needed for the purpose of grammar checking. Roughly, only 3,000 lines of code out of a total of 10,000 are required. If a UCP light is selected (based on these 3,000 lines of code) and rewritten in C, processing speed will increase substantially and the resulting pilot meet the needs for a commercialisation.

Size
Regardless of grammar checking alternative, size seems to present no problems for the Scarrie pilot.

Conclusions

- CORRie outperforms MorphoLogic as a spell checker for Scarrie.
- Both the CORRie fragment analysis approach and the ScarCheck approach outperform MorphoLogic as a grammar checking alternative for Scarrie.
- It is possible to integrate an external parser into the CORRie framework, and CORRie with two options for grammar checking, i.e. CORRie fragment analysis, and ScarCheck will provide the best platform for the Scarrie pilot. See proposed architecture below.

[1] Systematic tests on a large scale will be carried out in WP 7.

- Current implementation of CORRie fragment analysis is faster than the Lisp version of ScarCheck, and CORRie fragment analysis has a potential for covering more error types.

- Current implementation of ScarCheck for Swedish covers more error types than CORRie fragment analysis approach, and ScarCheck has a potential for speeding up (UCP light in C).

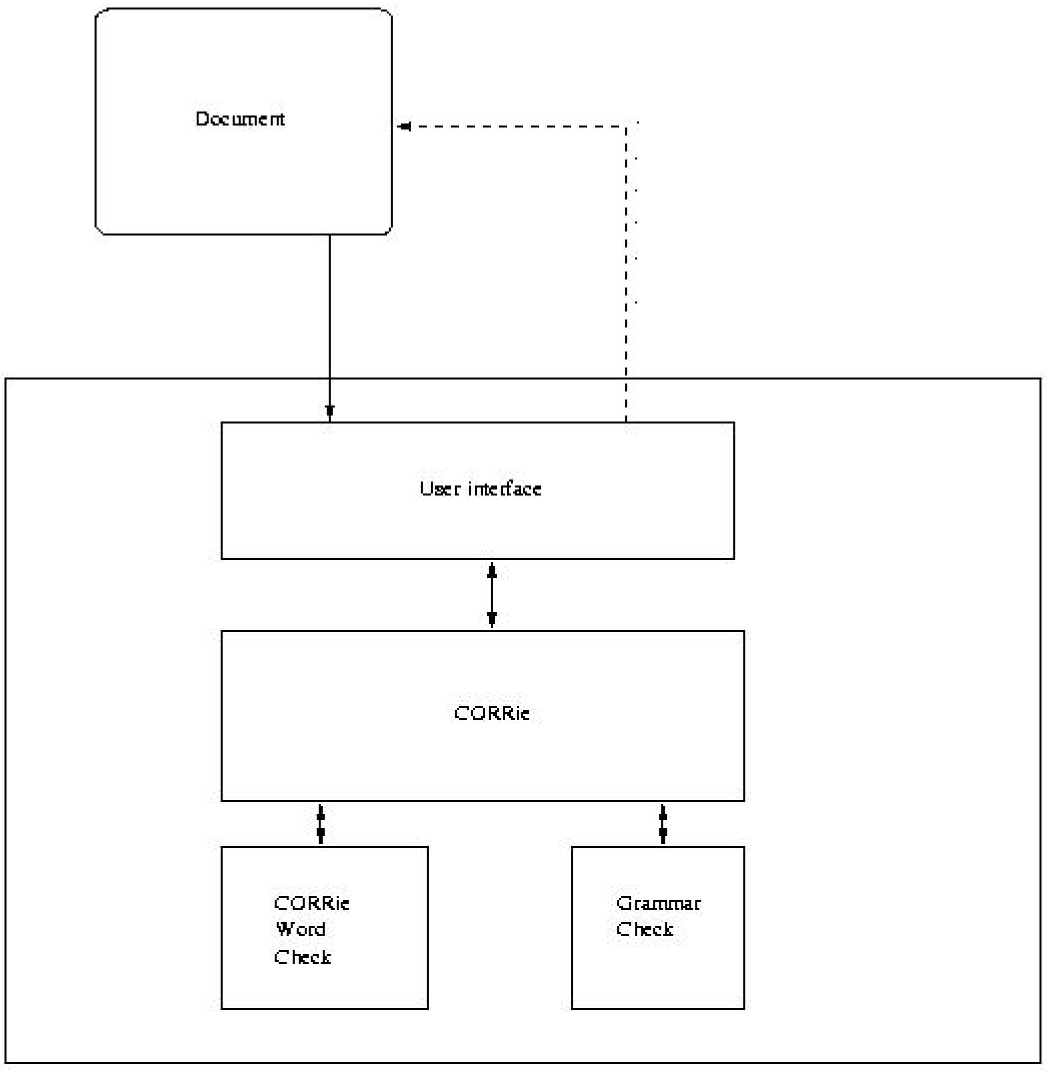
- Current demands on grammar checking functionality in terms of error coverage are found to be higher on the Swedish market than on the Danish and Norwegian markets.

  - Consequently, the UCP light version of ScarCheck will be the best option for Swedish.

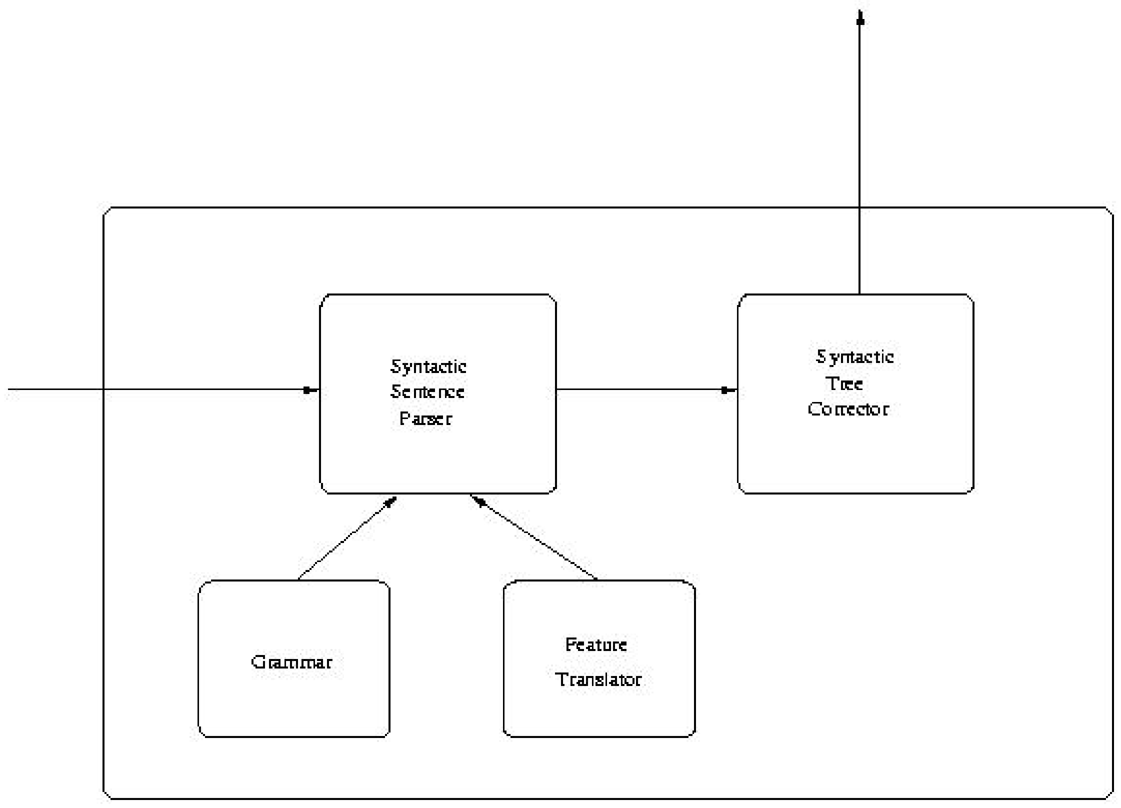
  - CORRie fragment analysis approach will be the best option for Danish and Norwegian.

Architecture

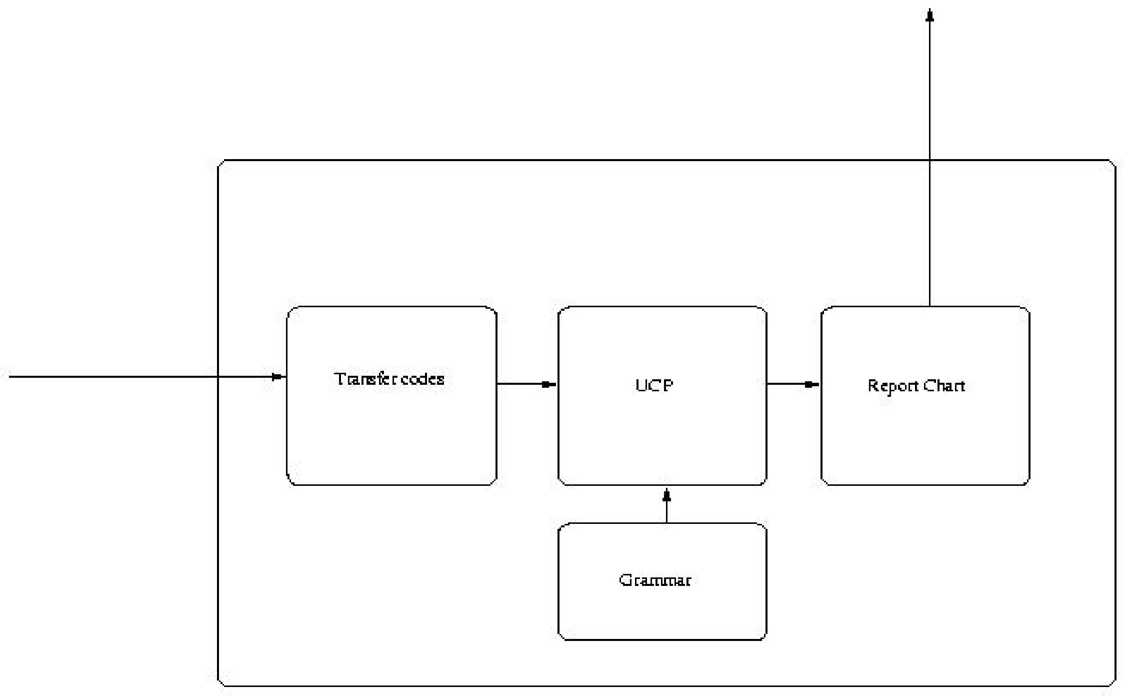
The Scarrie Pilot



Corrie Grammar Checker



ScarCheck Grammar Checker



References

CST 1998, *An inquiry on software for combined spelling and grammar checking. Product under consideration: CORRie***.** Scarrie common workspace. Technology baseline.

Music, Bradley, 1997, *Replacing the CORRie parser.* Scarrie common workspace.

Paggio, Patrizia, 1998, *Experiments with grammar writing in the CORRie formalism.* Scarrie common workspace, Work packages, WP6, Grammar testing, Danish.

Sågvall Hein, Anna, 1997. Input to the Scarrie grammar checking module. Scarrie common workspace, Work packages, WP6.

Sågvall Hein, Anna, 1998a, A chart-based framework for grammar checking. Initial studies. Proceedings of Nodalida '98, Copenhaguen 1998.

Sågvall Hein, Anna, 1998b, A test version of the grammar checker for Swedish. DEL 6.5.1. Scarrie common workspace, Work packages, WP6.

Sågvall Hein, Anna, Paggio**,** Patrizia and Wedbjer Rambell, Olga with contributions from Bart Jongejan, Leif-Jöran Olsson, Claus Povlsen, and Per Starbäck**,** 1998**,** *An inquiry on software for spell checking and grammar checking,* Scarrie common workspace, Technology baseline

Wedbjer Rambell, Olga, 1998, *A Minor Grammar Checking Test for Swedish Using the Fragment Analysis Approach in CORRie*, Scarrie common workspace, Work packages, WP 6, Grammar testing, Swedish.

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

Appendix I

Questionnaire for An inquiry of software for combined spell checking and grammar checking, 1st run
(The questionnaire was compiled by UU with input from CST and HIT).

1 I may provide a commercial software that performs spell checking and grammar checking.

2 It is robust and applies to unrestricted text.

3 Dictionaries and grammar are easily interchanged for different languages.

4 It uses the same dictionary for spell checking and grammar checking.

5 It recognises correct words that are not in the dictionary (by compound analysis and/or other means).

6 It suggests well-motivated corrections in a preferred order based e.g. on pronounciation, string similarity, and frequency.

7 The dictionary may include non-approved words and phrases, and suggest replacements.

8 It inserts hyphenation positions in accordance with markings in the dictionary.

9 It considers different style registers.

10 The dictionary may include multi-word expressions for correction of misspelled idioms and parsing efficiency.

11 a) Agreement errors in various phrase types (NPs etc.)

b) Erroneous verb sequences

c) Fundamental construction errors at clause level

12 Performance

(Feel free to state performance data in your own terms.)

The software runs on a computer with the following basic requirements:

…

It works with the following speed:
…

Appendix II

Monday, 18 May 1998

An inquiry on software for combined spell checking and grammar checking

by

Anna Sågvall Hein, Patrizia Paggio and Olga Wedbjer Rambell

with contributions from

Bart Jongejan, Leif-Jöran Olsson,

Claus Povlsen and Per Starbäck

Software alternatives under consideration:

1st alternative:
CORRie for spell checking and grammar checking
Main investigator: CST

2nd alternative:
CORRie for spell checking and ScarCheck for grammar checking (CO +SC)
Main investigator: UU

3rd alternative:
MorphoLogic for spell checking and grammar checking (MOR)
==

Answers to the questionnaire:

1 It is a commercial software that performs spell checking and grammar checking.

CORRie:	YES. (So far, the grammar checking part of the software has not been used in a commercial product.)
CO+SC:	YES. (So far, the grammar checking part of the software has not been used in a commercial product. Some work is needed to adapt it to such a use.)
MOR:	YES

==

2 It is robust and applies to unrestricted text.

CORRie: YES

It processes flat text with virtually unlimited line length. The program runs in a very stable way, which does not mean that it is without errors.

There are a few problems with the layout that can probably all be solved relatively easily:

- Output is written with a max column width that is defined as an input parameter (with a hard coded upper limit that can be easily changed, but necessitates recompilation). Input text with wider columns is wrapped between words - sometimes before punctuation, instead of after. There is no option to let the unbounded input column width survive in the output. Especially input text with no particular column width (e.g. text that only has new-line characters to denote the end of paragraphs) may appear with an unwanted layout in the output.

- An extra left-hand margin is added to the output. The margin contains blanks or the string --> to indicate an error. Errors are described on the same line (without indication in the margin) or on the next line (which starts with --> in the margin), depending on whether the text was wrapped or not.

There are also a few things that diminish CORRie's flexibility:

- Symbols below the ASCII value 32 seem all to be handled as white space. This may not always be desirable.

- The character set (e.g. Latin-1) of the input text must match the character set that is hard-coded in the program. The program cannot handle a text with more than one character set.

There is, at least in the Danish version, a bug in the system's handling of numbers.

Furthermore, CORRie does not always handle abbreviations correctly. The abbreviation "kr.", for example, is corrected to "c.".

CO+SC: YES.

MOR: YES

2.1 Do you cope with grammar checking without full parse?

CORRie: NO

If your answer is yes to 2.1, please, describe briefly the strategy you use

Otherwise, how is robustness ensured? Although a full sentence parse must be produced, rules may be written covering a sentence in fragments. Hereby it is possible to focus on the internal structure of certain syntactic elements leaving other elements unanalysed or unidentified. This has been done in the current Danish grammar. A detailed description can be found in the report “Experiments with grammar writing in the CORRie formalism” (available on the Scarrie workspace). This approach has also been tested for Swedish, and the results are presented in the report "A minor grammar checking test for Swedish using the fragment analysis approach in CORRie" (also available on the Scarrie common workspace). Furthermore, it should be noted that the system does not crash when a parse is not produced, and that spelling checking is performed anyway, so in this sense robustness is ensured.

CO+SC: YES

If your answer is yes to 2.1, please, describe briefly the strategy you are using

Robustness is ensured via partial parsing and the application of local error rules. By partial parsing we understand an approach where only certain types of constituents are analysed, such as NPs, PPs, APs, AdvPs, and VGs (the verbal core of the VP). The constituent analysis is also robust in itself in that it allows for feature relaxation for catching feature violations, such as agreement errors. Typically, there are no sentence rules. Segments that are not covered by grammar rules are stepped by in the analysis. As opposed to the fragmens in the CORRie fragment analysis, unanalysed segments need not be foreseen in the grammar.

The rules are formulated in a procedural formalism and invoked bottom-up at the recognition of lexical categories. For instance, the recognition of a determiner leads to the invocation of an NP-rule (designed for the recognition of NPs introduced by determiners). Local error rules are formulated in the same formalism as the grammar rules and invoked in the same manner. Whereas the partial parsing rules generate linguistic descriptions that may be used by other rules in the continued analysis, the application of the local error rules generates descriptions of erroneous fragments of constituents that are not to be used by other rules.

The ScarCheck machinery is implemented as two modules, a chart parser, UCP, and an error reporting module, REPORTCHART. The ScarCheck approach was presented at the NODALIDA 98: "A Chart-based Framework for Grammar Checking". It is available on the Scarrie common workspace as Del. 6.5.1b.

Otherwise, how is robustness ensured?

MOR: YES

First, the morphological analyser runs, and provides the next phase with all the information it can. The next phase is a sort of pattern matching using the above given morpho-syntactic symbols. Underspecification, that is, wildcards of different degrees, is allowed.

==

3 Dictionaries and grammars are easily interchanged for different languages.

CORRIE: Dictionaries and grammars themselves are easily interchanged for different languages. However, the program contains language-specific source code. Therefore, any executable version of the CORRie program is dedicated to a single language.

Examples of such language-specific data that had to be adapted to create the Danish version are:

- Binding morphemes and endings
- Placement of hyphens
- Character tables
- List of vowels
- Prefixes
- Features attached to the main grammatical categories
- Digits in words

The binary dictionaries and grammars must sometimes be recompiled after changes in the source code and cannot be used by versions of CORRie that are adapted to other languages

CO+SC: As regards the linguistics resources needed for spell checking by means of CORRie, see above. As regards the grammar, see 3.1 below.

MOR: YES, in principle, but "easily" is not a well-defined term.

3.1 Could you give a rough estimate of how much effort (in terms of pm) it would take to extend your software to a new language, i.e. adapting language resources such as:

grammar
dictionary

multi-word lists

compounding rules

character encodings

mark-up codes
pronunciation rules

provided that these resources are available in a machine-tractable form?

CORRIE: Adapting (converting) the CORRie platform to treat a new object language will approximately require two man days provided the linguistic resources are expressed in a formalism which CORRie can interpret. Information about how much manpower is needed to develop the various linguistic components (stated in the list) can be found in the Technical Annex of the SCARRIE project.

CO+SC: The CORRie + ScarCheck alternative includes a grammar for Swedish covering the fundamental error types that were identified in the error collection phase. Adapting this grammar for a new closely related language such as Danish or Norwegian should not require more than a couple of pw.

MOR: grammar:
rather patterns for typical potential erroneous structures
than grammar 8 pm

dictionary: words with morphological encoding 6 pm
(ca. 100,000 entries) (from scratch, but must be less becuase
of your existing sources)

compounding rules: (included in the morphological description)

character encodings no problem

pronunciation rules: unfortunately, we have not used pronunciation
rules yet but can be covered by the patterns, as well

3.2 What encoding format do the linguistic resources have (e.g. ascii, unicode)?

CORRIE: Latin-1

CO+SC: Latin-1

MOR: Not yet Unicode, but any 8-bit representation is usable.
===

4 It uses the same dictionary for spell checking and grammar checking.

CORRIE: YES

The same dictionary can be used for spelling and grammar checking provided the correct mapping between dictionary features and grammar features is specified in the relevant declaration. Dictionary and grammar features obey in fact different formats. The Danish grammar has been tested with a subset of the main dictionary to ensure that the feature mapping works correctly. In addition to the main dictionary, an exception dictionary can also be used to state additional lexical information to be used by the parser.

CO+SC: YES

MOR: YES

4.1 Is your (main) dictionary a full-form dictionary?

CORRie: YES

CO+SC: YES

MOR: NO, there are dictionaries for stems and affixes.

4.2 What kind of grammatical information is/can be included in the dictionary?

CORRIE: All kinds of grammatical information can be included in the dictionary

CO+SC: Information about word category and morpho-syntactic features is included in the Swedish dictionary together with information about grammar rules to be triggered. (So far, however, we found no way of including and accessing information about lemma and information holding for all the forms of a lemma in a convenient way, such as subcategorisation and semantic features.)

MOR: Depending on the language, but mainly the encoding of the morphemes' behavior before and after other morphemes.

4.3 Is there a limit to the number of grammatical features that may be included in the dictionary and used in the grammar checking process?

CORRIE: In principle, there is no limitation to the amount of grammatical information that can be included in the dictionary.

CO+SC: NO

MOR: NO, in principle there are no limits, in fact, of course, the program has some limits, but Hungarian morphology could also been described with it, so it must be enough for your languages, as well.

==

5 It recognises correct words that are not in the dictionary (by compound analysis and/or other means).

CORRie: YES
The Danish and the Swedish resources built so far include preliminary compound grammars that are used with reasonable success. For Danish, there are plans to add lexical restrictions concerning binding elements to obtain better precision results on compound analysis. Also the Swedish resource will be fine-tuned.

CO+SC: See CORRie.

MOR: YES

5.1 Does it perform capitalisation check?

CORRie: YES, mostly. If a word is coded in the dictionary with a capital letter, CORRie corrects it when it is spelt without capital. If a word is not spelt with a capital in the dictionary, on the other hand, CORRie does not correct it when spelt with a capital in the text.

CO+SC: See CORRie.

MOR: YES

5.2 Does it identify potential proper names?

CORRie: YES, although we haven’t tested this feature extensively, we have seen that the system recognises (at least some) potential proper names.

CO+SC: See CORRie.

MOR: There are proper names in the dictionary, but it does not identify new ones.

5.3 Does the system use other means for recognising unknown words?

CORRie: YES, frequency information is also used for recognising unknown words.

CO+SC: See CORRie.

MOR: The morphological analyzer we use is also the kernel module of our speller, so if it does not know a word, there are possibilities to add, as usual in spellers. There is a guesser in preparation that will help the user in adding linguistic info to the new words, automatically. Presently, we use a special user dictionary, called inflectional dictionary, where the user is expected to add two words per line: the unknown word in question and another one that behaves morphologically in a very similar way. It is a bit intuitive, we know, but it is the simplest way to provide the new words with linguistic information.

===

6 It suggests well-motivated corrections in a preferred order based e.g. on pronounciation, string similarity, and frequency.

CORRie: As regards word checking, the program can produce lists of corrections based on pronunciation, string similarity, and frequency. However, although the complete list can be inspected during development by way of a help program (ncorr-demo), only the highest scoring alternative is currently presented to the end user.

The grammar checker generates corrections for feature violation errors. It looks up the incorrect word in the dictionary and searches for an alternative word form of the same lemma with the correct set of features. If there are several candidates, the program chooses the alternative with the shortest edit distance to the erroneous word.

CO+SC: For word checking, see CORRie. For grammar checking, there is still no corrections mechanism available.

MOR: Well-motivated corrections, in a specific order (some sort of preference)

6.1 What’s the principle(s) for generating and ordering the corrections?

CORRie: Correction of spell checking errors in CORRie is based on the concept of 'minimal edit distance', which is defined as the number of changes needed to transform one word into another. When computing the minimal distance, CORRie both compares the orthographic strings corresponding to the invalid word under consideration and its possible replacement, and compares the phonetic representations of the same two words. The two scores obtained are totted up and used to pick the best possible replacement. Frequency information is also taken into account.

As regards correction of grammatical errors, see above.

CO+SC: See CORRie.

MOR: We have tried to collect most of the typical errors and rank them. So the system tries first to find patterns to the most critical errors, and so on.

==

7 The dictionary may include

a) non-approved words and phrases, and

b) suggest replacements of non-approved words

CORRie: YES, but phrases (both valid and invalid forms) are stored in a separate idiom list.

CO+SC: See CORRie.

MOR: YES

7.1 Does the software recognise and correct incorrectly split words?

CORRie: YES, if at least one of the segments resulting from the erroneous split is not itself a correct word to be found in the dictionary.

CO+SC: See CORRie.

MOR: YES

7.2 Does the software recognise and correct incorrectly joined words?

CORRie: YES (according to system documentation, not confirmed by preliminary testing, but no attempts made to investigate the cause of failure)

CO+SC: See CORRie.

MOR: YES, what can be recognised on a formal basis. (There is no semantics.)

7.3 Does the software recognise repeated words?

CORRIE:	YES
CO+SC:	See CORRie.
MOR:	NO, but it can be added easily (word-processors calling our functions do it without linguistics).

If the answer is YES, how many words can the repetition consist of?

CORRie:	According to system documentation, 16. We have been able to obtain recognition of a repetition consisting of 3 words within the same sentence.
CO+SC:	See CORRie.

==

8 It inserts hyphenation positions in accordance with markings in the dictionary.

CORRie:	It should be possible to insert hyphenation positions in the dictionary entries and have the search process ignore them.
CO+SC:	See CORRie.
MOR:	YES, it offers the hyphenation positions, and inserts them if needed.

8.1	Are the suggestions for hyphenation positions stored with the dictionary entries or calculated by means of rules during the processing?
CORRie:	Stored with the dictionary entries.
CO+SC:	See CORRie.
MOR:	Both.

==

9 It considers different style registers.

CORRie: YES.
The vast majority of commercial style checkers available consider only isolated words in order to distinguish between writing styles. This form of style checking can be done in the CORRie platform. The dictionary format in fact allows for tagging of words in the dictionary to express that
1) the word is only valid in the current style and is otherwise rejected
or that
2) the word will only be replaced under a certain style and accepted otherwise

CO+SC: See CORRie.

MOR: YES

9.1 How many different style registers may it consider?

CORRie: For each type of tagging it is possible to express 7 different styles.

CO+SC: See CORRie.

MOR: In this moment, three. It can be changed, of course, if there is a reason why.

9.2. Is a style register consistently enforced throughout the document?

CORRie: YES, if the coding in the dictionary is consistent

Since CORRie already makes statistics of the input document (e.g. average sentence length) it should be possible to add information on the writing style to these statistics for the user to inspect.

CO+SC: See CORRie.

MOR: It is a question of the calling module, not the grammar checker itself.

==

10 The dictionary may include multi-word expressions for correction of
misspelled idioms and parsing efficiency.

CORRie: YES, multi-word idioms and their possible misspellings are stored in a separate idiom list. Misspelt idioms can be corrected if the error is foreseen, i.e. an invalid form. In addition, an idiom may be identified as incorrect (but with no correction generated) if any of the in-going words is missing in the word form dictionary.

CO+SC: See CORRie.

MOR: Partly solved.

10.1 Does the dictionary include misspelled idioms with suggestions for corrections?

CORRie: YES, the idiom list can include invalid forms.

CO+SC: See CORRie.

MOR: YES, it is possible to do.

10.2	Does the dictionary include multiword expressions for parsing efficiency, and may these expressions be assigned information about word category and other kinds of features?
CORRie:	Experiments with the way in which the idiom list interacts with the parser is carried out for Swedish. According to the documentation, idioms can be treated by the parser as sequences of independent words or as units, depending on a code attached to each idiom in the idiom list. Category and features can be assigned in the idiom list. The format used is the same as in the exception dictionary.
CO+SC:	See CORRie.
MOR:	Under development.

==

11 It handles systematic grammatical errors such as

a) Agreement errors in various phrase types (NPs etc.)
b) Erroneous verb sequences
c) Fundamental construction errors at clause level

CORRie:	So far, only agreement errors have been treated in the fragment analysis approach. Every sentence is assigned a full parse in terms of recognised fragments. So far, only one sentence rule has been used, both in the Danish and the Swedish test grammar. The sentence rule expands into fragments. A fragment may be an NP, and then there is a number of rules determining what an NP may look like. A fragment rule expands into a phrase type (more or less completely described by usual rules or local error rules) or into a terminal symbol (matches the word and makes no attempt at building a phrase). The fundamental question in determining the potential of this approach is to decide how much has to be specified in the grammar, i.e. how comprehensive it must be in terms of sentence rules, clause rules etc. in order to recognise the errors that the pilot should focus on. So far, no attempt has been made at formulating clause rules. In other words, how complete must the grammar to be in order to capture the fundamental errors without generating false alarms?
CO+SC:	YES, the grammar comprises rules for the three types of errors.
MOR:	Most but not all of them are covered.

11.1	Is the method used for grammar checking based on statistics or grammar rules?
CORRie:	Grammar rules
CO+SC:	Grammar rules
MOR:	Local grammar rules.

If your system uses a grammar, can you give an example of a simple grammar rule for the treatment of agreement?

CORRie: Examples are shown in the report “Experiments with grammar writing in the CORRie formalism”, and in the report "A minor grammar checking test for Swedish using the fragment analysis approach in CORRie."

CO+SC: YES.
Below you will find an example of a grammar rule for the treatment of agreement errors in NPs. The rule is designed for the recognition of the head noun in NPs consisting of a determiner, an adjective phrase, and a noun. Three kinds of agreement errors are captured, i.e. violation of number agreement, gender agreement and species (form) agreement between the premodifier (determiner and adjective phrase) and the head noun. A description of the phrase thus recognised is stored in the chart to be used in the further processing. Further, the recognition of a following relative clause (det.rel.tail) is initiated. This rule invocation is conditioned by the determiner; it has to be a definite article. The application of the subrule det.rel.tail imposes constraints on the species of the head noun when modified by a relative clause. Finally, if the NP is in the genitive case, processing for an NP introduced by a genitive attribute is invoked. Error features are assigned values in accordance with Error typology for automaatic proof.reading purposes (see DEL 2.1).

```
(define sve.gram-entry np.det.adjp_noun
#u <* word.cat> = 'noun,
(<& numb>:=:<* numb>/<& err :new>:=:'gpnpag01),
(<& gender>:=:<* gender>/<& err :new>:=:'gpnpag02),
(<& form> :=: <* form>/<& err :new>:=:'gpnpag03),
<& case>:=:<* case>,
<& head word.cat> :=:<* word.cat>,store,
(<& det word.cat>='art,not <& err :last>='gpnpag01,
advance(det.rel.tail)/continue),
(<& case>:=:'gen,assign.majorprocess(np_poss)/continue);
#!         word.cat;)
```

MOR: Agreement is not a critical problem in Hungarian, so it is not yet included, but we can describe the agreement problem for other languages with the help of the present formalism.

11.2 Is there a limit to how many tokens an identifiable error can span over?

CORRie: In principle, NO.

CO+SC: NO

MOR: In principle no, in fact, we have never tried to exceed 8-9 tokens.

11.3 What types of agreement errors are recognised?

CORRie: Currently, the Danish grammar contains rules for the treatment of gender, number and definiteness agreement in NPs of limited comnplexity. It could easily be extended to treating e.g. subject predicate agreement.

CO+SC: Currently, the Swedish grammar covers the following types of agreement errors, classified in accordance with the error typology (DEL 2.1).

```
GPNPAG01 "number agreement in premodifier - noun"
GPNPAG02 "gender agreement in premodifier - noun"
GPNPAG03 "species agreement in premodifier - noun"
GPNPAG04 "definite noun form instead of indefinite"
GPNPAG08 "number agreement in noun with apposition"
GPNPAG13 "gender agreement in premodifier - pnoun"
GPAPAG01 "agreement in coordinated adjective phrases"
GPAPAG02 "agreement in parallel adjective phrases"
```

MOR: --

11.4 What types of errors are recognised in the verb phrase?

CORRie: In principle any type that can be captured by way of a rewrite rule (either a 'normal' or an error rule in terms of the CORRie formalism). However, only some types of errors in the verb phrase are domain-revelant (subcategorisation errors, for example, are relatively rare in the Danish corpus).

CO+SC: Currently, the grammar handles the following types of verb phrase errors:

```
GPVFFV01 "infinite verb instead of finite"
GPVFMF01 "doubled verb in the finite form"
GPVFMF05 "supine form instead of imperative"
GPVFMV04 "past tense + past tense" => past tense + infinitive"
```

MOR: E.g. missing argument, in some special cases.

11.5 What types of errors are recognised at clause level?
We think of errors such as missing finite verb:
* “Man kanske inte behov av ….” -> Man kanske inte har behov av ….”.

CORRie: We have not experimented with errors at clause level. Some of them should not be too difficult to treat, e.g. the one mentioned, which could be treated by a sentence error rule where none of the fragments the sentence consists of is a finite verb.

Errors the treatment of which presupposes a complete and meaningful parse of the whole sentence, however, may be too costly: an example relevant to Danish could be incorrect word order in subordinate sentences. Such an error, however, is quite rare in our domain. Therefore, we are not planning to handle it.

CO+SC: Currently, the grammar handles the following type of error at clause level:

`GPVVMV01 "finite verb missing"`

MOR: Missing finite verb, exactly, or more than one verb without conjunctive elements in between, etc.

11.6 What other types of errors are recognised?

CORRIE: CORRie has a facility for splitting run-ons and joining incorrectly split words; we hope to be able to make these routines interact with the grammar so that in cases where both possibilities are valid in isolation (as often the case), the correct alternative is made to depend on the grammatical context.

CO+SC: `GPPCOF01 "subjective form => objective form"`

MOR: Missing obligatory elements. e.g. prepositions (postpositions, in fact, in Hungarian) without reference.

11.7 Does the system generate suggestions for corrections for the grammatical errors that it identifies?

CORRIE: YES, if the error is recognised as a feature violation error and
an alternative with the correct features can be found in the dictionary.

CO+SC: Not in its current version. It can be extended to do so, however, partly falling back on CORRie's correction mechanisms.

MOR: Yes, in some cases, but morphologically they are not always perfect. In spite of our existing inflectional thesaurus module that uses generation tools, here in the grammar system we have not used it yet.

11.8 Does the system generate diagnoses for the grammatical errors that it idenfies? If, YES, please given an example.

CORRIE: YES,
if an error rule has been applied (see the report for more details and examples).

CO+SC: It depends on how "diagnosis" is understood. The recognition message currently generated by the system is formulated in accordance with the quite elaborate four level error typology defined in the project (see DEL 2.1).

MOR: It quotes the Orthographical Advisory Dictionary's adequate paragraphs as diagnosis.

===

12 PERFORMANCE

CO+SC: CO+SC is the result of an integration of CORRie and ScarCheck. The two modules are linked via an interface that was developed by UU. CORRie handles word checking and ScarCheck grammar checking. The two modules use a common dictionary, and dictionary search is carried out once for each word type.

As a result of the word checking process syntactic codes are assigned to the word types in CORRie (retrieved from the dictionary or generated by rules), and these codes are forwarded to ScarCheck for grammar checking. ScarCheck(2) translates the codes into AV-structures and builds a chart of these structures. Thus the initial chart will comprise word edges only, as opposed to character edges as in the original version of ScarCheck. This simplification (no dictionary search, initial chart of word edges, no character edges) reduces processing time by almost 50%.

ScarCheck consists of two modules, a chart parser, UCP, and a chart interpreter, Reportchart. They are both written in Commonlisp There are several implementations of Commonlisp. Scarcheck uses CLISP which is a free ware that runs under DOS, OS/2, Windows NT, Windows 95, Amiga 500-4000, Acorn RISC PC, and Unix. In principle, thus, UCP runs on all these platforms eventhough some minor modifications may be needed when changing platforms e.g. because of different file systems.

The program scarcheck(2) is a "memory dump" of a lisp comprising UCP, Reportchart and the grammar that is to be used. Current version of scarcheck(2) with a grammar of 41 rules occupies approximately 1,3 Mbyte disk space. This dump and CLISP is what is needed to run the checker.

All in all, there are 218 different syntactic codes that are transferred from CORRie to UCP.

MOR: The software runs on a computer with the following basic requirements:
The kernel part of or software has been written in standard C and C++,
the code is portable and can be compiled by most of the
well-known C and C++ compilers.

It works with the following speed:
Difficult to give concrete data, but the general speed must be usable, because quite a lot of international companies (Microsoft, Lotus, Inso, Franklin, Proximity, Rank Xerox, etc.) have licensed our proofing tools.

12.1 Is there a demo version of your system for public testing?

CORRIE: The project has several running versions available.

CO+SC: There is one version available for testing the integrated alternative (with the complete Swedish dictionary), and two versions available for testing ScarCheck in isolation. The test versions are currently available at the Unix systems at UU, but interested parties can get guest accounts there to try them out. You will need an Internet connection and an X server (for instance, Exodus for Windows.

MOR: NO, but I offer the full version for testing purposes (it needs someone who knows some Hungarian...).

Please, provide figures concerning performance on a typical document (specify number of running words, syntactic complexity, number of error tokens, number of error types, and platform)

CORRie: On a unedited domain-relevant document run on an HP 9000 (170 MHz), with the parser active, CORRie generated the following statistics:

CPU time: 88 sec.
Elapsed time: 0:57
2904 words, 1249 unique
143 sentences, with an average of 20.31 words per sentence
Gunning-Fog: 17.4
Flesch: 29.2
Flesch-Kincaid: 14.2
Raygor Readability Estimate: Coll

Please, provide the following figures obtained on a typical document:

- recall (number of valid words recognised/total number of valid words, and number of errors flagged/total number of errors)
- precision (number of correct flaggings/number of flaggings)
- suggestion adequacy (number of correct first suggestions/number of flaggings)

CORRIE: On the same document, the following results were obtained:

Recall

2904 total words

2800 valid words

2645 (94.5%) valid words accepted

155 (5.5%) valid words rejected (bad flags)

104 invalid words (real errors)

32 (30.8%) real errors spotted (good flags)

72 (69.2%) real errors missed

Precision

187 flaggings

32 (17.1%) good flags

155 (82.9%) bad flags (false positives)

Suggestion adequacy

32 good flags

4 (12.5%) hits on initial suggestion

0 (0.0%) hits on non-initial suggestion

14 (43.8%) misses (suggestions offered, none correct)

14 (43.8%) with no suggestions offered

A brief analysis of the output showed that:

- bad flags are mostly due to
    - the bug in the treatment of numbers
    - some unrecognised proper names (probably due to the fact that the text was relatively short)
    - unrecognised foreign words
- misses are mostly due to:
    - punctuation errors
    - false negatives due to agreement
    - binding elements in compounds
- no replies are mostly due to:
    - capitalisation errors
    - binding elements in compounds

In general, the figures obtained may seem rather poor. However, they reflect a rather tough evaluation methodology, as even a missing comma in the CORRie output counts as an error when the output is compared with the corresponding proof-read version. Furthermore, it must be remembered that the dictionary is the only component of the Danish version of the system that can be considered complete at this stage.

CO+SC: So far, only limited testing has been performed with the integrated

alternative. The object of the test was a demo text consisting of 346 tokens. The complete Swedish dictionary and the complete grammar was engaged. Processing time compared to the CORRie alternative is approximately, 3.25 times slower. The test focused on grammatical errors and they were identified with the same precision and recall as in ScarCheck in isolation.

The integration work and subsequent systematic testing was for quite some time hampered by a serious bug in CORRie with the effect that syntactic codes were not generated for all the dictionary alternatives and could thus not be forwarded to the grammar checker. This problem and several others have now been solved and systematic testing may be performed.

In order to turn the combined CORRie+ScarCheck alternative into a commercial product a light version of UCP containing only the relevant parts for this application should be defined and rewritten in the C programming language. Meanwhile, for testing and functional validation, the current version of the checker is appropriate.

MOR: NO ANSWER

Appendix III

UU/Leif-Jöran Olsson

The integration of ScarCheck into CORRie

1. Checked the documentation "How to integrate a parser into CORRie" to find out which functions and datatypes that were involved in the grammarcheck.

2. Added the expect library to the Makefile, Corrie2.c and lrparse.c

3. Completely rewrote the function CheckSentence and called it CheckSentence2 to send the transfercodes to ScarCheck. Adds transfercodes to interpunctuation.

4. Received bugfix from Vosse for the function DetermineNewWords, to get at all the transfercodes for each word.

5. Moved MakeExtraInfo to gramlexint.c and changed the order of includefiles in associated files.

6. Changed DetermineAppearance2 to call LookUpWord2 if not LookUpWord. Adds proper noun transfercodes.

7. Wrote new LookUpWord called LookUpWord2, which returns ExtraInfoPtr instead of CATREP.

8. Wrote new FilterCompoundCat called FilterCompoundCat2, which returns ExtraInfoPtr instead of CATREP.

9. Wrote new AddPrivate called AddPrivate2, which handles ExtraInfoPtr instead of CATREP.

10. Wrote new ConvertString called ConvertString2, which returns ExtraInfoPtr instead of CATREP. Added calls in gramlexint.c, idiomint.c and Corrie2.c.

11. Wrote new ConvertDict called ConvertDict2, which handles ExtraInfoPtr instead of CATREP. Added calls in gramlexint.c, lrparse.c and Corrie2.c.

12. Added new struct DictTree2 and type DICTTREE2, which handles ExtraInfoPtr instead of CATREP.

13. Conditioned block in DetermineAppearance2 with check for ScarCheckP.

14. Conditioned block in SetUpDictionaryAndGrammar with check for ScarCheckP. Added wildCard2 which is ExtraInfoPtr instead of CATREP.

15. Conditioned block in main with check for ScarCheckP.

16. Conditioned blocks in ReadUsersDictionary with check for ScarCheckP.

17. Wrote new LookUpPrivate called LookUpPrivate2, which returns DICTTREE2 instead of DICTTREE.

18. Wrote new type IdiomRepPtr2 and struct IdiomRep2, which handles ExtraInfoPtr instead of CATREP.

19. Wrote new WordDefinition called WordDefinition2, which looks up ExtraInfoPtr instead of CATREP via LookUpWord2.

20. Wrote new MakeSentence called MakeSentence2, which handles ExtraInfoPtr instead of CATREP.

21. Wrote new type SENTENCE2 and struct Sentence2, which handles ExtraInfoPtr instead of CATREP.

22. Wrote new PutInCache called PutInCache2, which handles ExtraInfoPtr instead of CATREP.

23. Changing calls to MemberOf in new functions to existing WrdNCmp, which

handles ExtraInfoPtr (chars) instead of CATREP.

24. Moved MakeExtraInfo and struct ExtraInfoPtr again (see 5) from gramlexint.c to lrparse.c and changed the order of includefiles gramlexint.h and lrparse.h in lrparse.c.

25. Wrote new IdiomRep called IdiomRep2, to handle idioms, which handles ExtraInfoPtr instead of CATREP.

26. Wrote new BeginIdiom called BeginIdiom2, to handle idioms, which handles SENTENCE2 instead of SENTENCE.

27. Wrote new TryBeginIdiom called TryBeginIdiom2, to handle idioms, which handles SENTENCE2 instead of SENTENCE.

28. Wrote new TryIdiomRep called TryIdiomRep2, to handle idioms, which handles SENTENCE2 instead of SENTENCE.

29. Wrote new IsANoun called IsANoun2, to handle idioms, which handles SENTENCE2 instead of SENTENCE.

30. Conditioned blocks in InitIdiom with check for ScarCheckP. Added concatClass2 which is ExtraInfoPtr instead of CATREP.

31. Wrote new ConvertSentence called ConvertSentence2 to handle idioms, which handles ExtraInfoPtr instead of CATREP. (called in ParseErrorModeInit, which is not present in CheckSentence2)

32. Conditioned blocks in ProcessSentenceElement with check for ScarCheckP.

33. Added idiom handling to CheckSentence2.

Appendix IV

Examples of input to ScarCheck from CORRie, and results of the processing

The sentences are run in pairs, right version followed by wrong version. They are presented after the results of the processing. When reportchart gives no error message, the sentence has been found correct.

```
(sp '((PNXPS ALXPD ) (NLOXB ) (NNUPDB ) (PR ABX ) (PMNX ) (PCPUSDG VBPRM )
(NLCXX ) (PUNC) ))

> (reportchart)

>     De första medlingarna i Västerås genomfördes 1994.

==========================================

(sp '((PNNSZ NNNSIB ALNSD ) (NLOXB ) (NNUPDB ) (PR ABX ) (PMNX ) (PCPUSDG
VBPRM ) (NLCXX ) (PUNC) ))

> (reportchart)

INTERVALL: 1,3

FEL: number agreement in premodifier - noun

>     Det första medlingarna i Västerås genomfördes 1994.

==================================================

(sp '((NNNXIB ) (VBARM PCPXSDB ) (ABX ) (PR ABX ) (PNXPS ALXPD ) (AVXXXBC
) (NNUPDB ) (CN ) (NNUPDB ) ))

> (reportchart)

> Folk väntade förmodligen på de större maskinerna och traktorerna

================================================

(sp '((NNNXIB ) (VBARM PCPXSDB ) (ABX ) (PR ABX ) (PNNSZ NNNSIB ALNSD )
(AVXXXBC ) (NNUPDB ) (CN ) (NNUPDB ) ))

> (reportchart)

INTERVALL: 5,7

FEL: number agreement in premodifier - noun

>     Folk väntade förmodligen på det större maskinerna och traktorerna

================================================

(sp '((PNNSZ NNNSIB ALNSD ) (VBAPC ) (AVNSIBP ) (SNO IE ) (VBAIM ) (PR ABX )
(AVZZZBP ) (NNUPIB ) (PUNC) ))

> (reportchart)

>     Det är nödvändigt att tänka i nya banor.

===================================================
(sp '((PNNSZ NNNSIB ALNSD ) (AVNSIBP ) (SNO IE ) (VBAIM ) (PR ABX ) (AVZZZBP
) (NNUPIB ) (PUNC) ))
> (reportchart)
INTERVALL: 1,2
FEL: finite verb missing
>     Det nödvändigt att tänka i nya banor.
================================================
****************
Compounds:
----------------
anläggningskostnader
mittfältsstrateg
```