An object-oriented rule language for high-level text processing

Ola Knutsson, Johan Carlberger and Viggo Kann
Department of Numerical Analysis and Computer Science
Royal Institute of Technology
SE-100 44 Stockholm, Sweden
{knutsson, jfc, viggo}@nada.kth.se
www.nada.kth.se/theory/projects/granska/

Background

An object-oriented rule language was developed for the application of a grammar-checker for Swedish, Granska (Domeij et al, 2000). However, during the development of the rule language, it has evolved to a more general rule language for high-level text processing. There were several objectives for the development process; but the main was to design a rule language which is easy to use and powerful and that is still robust and efficient. In addition, maybe the most important objective: we wanted to design and implement our own rule language for full control and possibilities of extensions and experiments.

Object-orientation and linguistic power

The rule language is object-oriented and has a syntax resembling Java or C++. The linguistic expressive power is influenced by Constraint Grammar (Karlsson, 1995) and languages for Finite State parsing (see for example Karttunen et al, 1996). However, the rule language differs in its possibilities for higher linguistic abstraction given by phrase structure rules with features and values.

Rules for surface syntax analysis

Rules for grammar-checking have already been presented in (Domeij et al, 2000; Knutsson, 2001) and therefore more general parts of the rule language are presented. The rules for general analysis are called help rules and can be used as subroutines from other rules. A help rule that detects Swedish noun phrases like en bil, den röda bilen and några bilar is shown in Rule 1 below. The rule is on the form NP --> Determiner (Adjective) Noun with internal agreement in gender, number and species.

Rule 1:

```
NPmin@     {
    X(wordcl=dt),
    Y(wordcl=jj & gender=X.gender & num=X.num &
    spec=X.spec)*,
    Z(wordcl=nn & gender=X.gender & num=X.num &
    spec=X.spec)
-->

    action(help)              }
```

Rule 1 has two parts separated with -->. The first part is a matching condition and contains the head of the rule, NPmin, and the rule body given by the variables X, Y and Z. X contains the determiner and Z specifies that the determiner should be followed by a noun (possibly preceded by one or more (*) adjectives), which agrees in gender (gender=X.gender), number (num=X.num) and species (spec=X.spec). It is also possible to define recursive rules, which is necessary for an efficient implementation of for example Swedish noun phrases.
With the operator ; which means logical or between rules, it is possible to define a union of rules, for example several

Rule 2:

```
NP@          {
(Npmin) () --> action(help);
…;
(NpwPP) () --> action(help)

}
```

By this construction, it is possible to add increasingly linguistic information with only minor changes in the rule code. This is important for the development process and helps the grammarian to keep control over the rule collection.

The rule language has first of all been applied to grammar-checking, but also with some success to the following areas:

- Tag correction, rules for correction of wrong part-of-speech tagging. Tag correction can be done directly with help rules, which reassign words, with a new correct tag.
- Clause boundary detection, rules for detection of clause boundaries, see example below.
- Noun phrase recognition, rules for recognition of Swedish noun phrases. Some examples of the NP-recognition will be presented below.
- Transformations, rules for inflection and transformation of words, phrases and clauses. This has only been tried out experimentally, and will be further explored in the near future.
- Syntactic functions, when we are satisfied with the flat phrase structure analysis in Granska, we will start work with syntactic functions.

These five applications will improve the grammar-checking in Granska and will in the near future be used in the area of text summarization.
The rule language is carefully implemented using yacc, flex and C++. The rule syntax can easily be extended and new methods can be implemented without much effort. The rule matcher is optimized with statistical means.
The whole Granska system (with about 20 rule categories and 250 error rules) processes about 3 500 words per second on a SUN Sparc station Ultra 5, tagging included. The numbers are hard to compare, but we believe that we have achieved a comparably high performance.

Clause boundary rules

Clause boundaries are important for all kind of syntactic analysis. One of the clause boundary recognition rules, is like the following below: it states that there is a clause boundary if a conjunction (Y) is preceded and followed by a pronoun, noun, proper noun or an adverb. The rule is inspired by Ejerhed's algorithm for finite state segmentation of discourse into clauses (Ejerhed, 1999). The rule is context-sensitive which means that only the conjunction is matched as a clause boundary delimiter by other rules. In this way, words or sequences of words can be assigned with new labels like for example clause boundary delimiters, syntactic functions or conjunction types.

```
cl_del@      {
V(sed!=sen),
X((wordcl=pn & pnf=sub)|
(wordcl=pm & case=nom) |
(wordcl=nn & case=nom) |
wordcl=ab),
ENDLEFTCONTEXT,
Y(wordcl=kn),
BEGINRIGHTCONTEXT,
Y2(((wordcl=pn & pnf=sub) |
(wordcl=pm & case=nom) |
(wordcl=nn & case=nom) |
wordcl=ab) & wordcl=X.wordcl),
Z(wordcl=vb & (vbf=prs | vbf=prt | vbf=imp))
-->
action(help, text:=Y.text,wordcl:=Y.wordcl)

}
```

Example output from Granska's rule matcher

All NPs, PPs and other constructions that Granska recognizes will not be presented here; instead, the analysis of two simple sentences will presented with output from Granska's rule matcher. The morfosyntactic values of a word or a phrase are given in column 3. Column 4 presents the regent rule that has been applied. Many rules need other help rules for detection; however, the help rules are only applied if necessary. The sentence examples below are taken from Källgren (1992). The output format from the rule matcher is "work in progress".
The first example shows that only the "best" NP-candidates are presented. These choices are mainly done with heuristics and longest matchings.

Input: *Hunden eller katten äter fisken och köttet*.
(The dog or the car eats the fish and the meat)
Output:

```
0      $                      <sen clb>                (clbegin)
1-3    Hunden eller katten    <nn utr sin def nom>     (np_comp)
4      äter                   <vb prs akt>             (vbchain4)
5-7    fisken och köttet      <nn utr plu def nom>     (np_comp)
8      .                      <mad cle>                (clbegin)
```

The second example gives a problem of coordination. The conjunction can operate between NPs or between clauses. The rules for clause boundary recognition (clbegin and clend) detect that the conjunction *och* is coordinating two clause and not the two NPs *köttet* and *katten*. All NPs in the sentence are minimal and contain only a definite noun.

Input: *Hunden äter köttet och katten äter fisken*.
(The dog eats the meat and the cat eats the fish)
Output:

```
0      $           <sen clb>                (clbegin)
1      Hunden      <nn utr sin def nom>     (np_min)
2      äter        <vb prs akt>             (vbchain4)
3      köttet      <nn neu sin def nom>     (np_min)
                   <cle>                    (clend)
4      och         <kn clb>                 (clbegin)
5      katten      <nn utr sin def nom>     (np_min)
6      äter        <vb prs akt>             (vbchain4)
7      fisken      <nn utr sin def nom>     (np_min)
8      .          <mad cle>                 (clend)
```

References

Domeij, R., Knutsson, O., Carlberger, J. & Kann, V. (2000). Granska – an efficient hybrid system for Swedish grammar checking. I Proc. 12th Nordic Conference in Computational Linguistics, Nodalida-99. Department of Linguistics, Norwegian University of Science and Technology, Trondheim, pp. 49-56.

Ejerhed, E., Källgren, G., Wennstedt, O. & Åström M. (1992) The Linguistic annotation system of the Stockholm-Umeå Corpus project. Technical Report DGL-UUM-R-33, Department of General Linguistics, University of Umeå, Umeå, Sweden.

Ejerhed, E. (1999). Finite state segmentation of discourse into clauses. I A. Kornai, editor, Extended Finite State Models of Language. Cambridge University Press, chapter 13. Cambridge.

Karlsson, F. (1995). The Formalism and Environment of Constraint Grammar Parsing. In Karlsson, F. Voutilainen, A. Heikkilä, J. & Anttila, A. (eds.). Constraint Grammar. A Language Independent System for Parsing Unrestricted Text, Mouton de Gruyter, Berlin, Germany.

Karttunen, L. Chanod, J.P. Grefenstette, G. Schiller, A. (1996) Regular Expressions for Language Engineering, in Natural Language Engineering 2 (4) pp. 305-328.

Knutsson, O. (2001) Automatisk språkgranskning av svensk text. Licentiatavhandling, TRITA-NA-01-5, ISBN 91-7283-052-2, Institutionen för numerisk analys och datalogi, Kungliga Tekniska Högskolan, Stockholm.

Källgren, G. (1992). Making maximal use of morphology in large-scale parsing: the MorP parser. PILUS 60, Department of Linguistics, Stockholm University, Stockholm.