

Interactive pedagogical programs based on constraint grammar

Lene Antonsen, Saara Huhmarniemi, Trond Trosterud

<http://oahpa.uit.no>

OAHPA!



Abstract

We here present a set of interactive parser-based CALL programs for North Sámi. The programs are based on a finite state morphological analyser and a constraint grammar parser which is used for syntactic analysis and navigating in the dialogues. The analysers provide effective and reliable handling of a wide variety of user input. In addition, relaxation of the grammatical analysis of the user input enables locating grammatical errors and reacting to the errors with appropriate feedback messages.



Background and pedagogical motivation

The pedagogical programs in OAHPA! are based upon three pre-existing language technology resources developed at the University of Tromsø: a morphological analyser/generator, a CG parser for North Sámi and a number word generator compiled with the Xerox compiler xfst.

The main goal of the development of OAHPA! was to develop a language tutoring system going beyond simple multiple-choice questions or string matching algorithms, with free-form dialogues and sophisticated error analysis. Immediate error feedback and advice about morphology and grammar were seen as important requirements for the program.

Pedagogical lexicon

The OAHPA! programs share a set of common resources: a pedagogical lexicon and a morphological generator that is used for generating the different word forms that appear in the programs. The dialectal variation is taken into account in the lexicon as well as in the morphology. The lexical entry for *monni* "egg" is given below:

```
<entry>
<lemma>monni</lemma>
<pos class="N"/>
<translations>
<tr xml:lang="nob">egg</tr>
<tr xml:lang="fin">muna</tr>
</translations>
<semantics>
<sem class="FOOD-GROCERY"/>
</semantics>
<stem class="bisyllabic" diphthong="no"
gradation="yes" soggi="i" rime="0"/>
<dialect class="NOT-KJ"/>
<sources>
<book name="d1"/>
<book name="sara"/>
<book name="algu"/>
</sources>
</entry>
```

The morphological properties of words are used when making a detailed feedback on morphological errors. If the user does not inflect the lemma correctly, she can ask for hints about the inflection, and try once more, instead of getting the correct answer straight away.

Training numbers

The Numra game offers training on number expressions, either from number symbols to linguistic representation, or vice versa. As our only program, Numra offers training for four different Sámi languages, Kildin, Inari, North, Lule and South Sámi. We hope to extend our coverage to more languages for the other programs as well.

Morphological feedback

The feedback messages are determined by the combination of morphological features in the lexicon and the inflection task at hand. The morphological specification below gives a rule stating that there is a vowel change in illative singular for bisyllabic nouns that end with the vowel *i*. The corresponding feedback message instructs the user to remember the vowel change.

The system-internal representation of *monni* states it is a bisyllabic i-stem, which triggers *i > á* change in illative.

monni
monni
*monni has even-syllabic stem and shall have strong grade. Vowel change i > á, the suffix -i.
monni
monnái

The user types the erroneous *monni*, and gets feedback from the machine. A correct answer gets green colour.

```
<stem class="bisyllabic" soggi="i">
<msg case="Ill" number="Sg">L.á</msg>
</stem>
<message id="L.á">Vowel change i > á.
</message>
```

+A+Comp: iX4b BUSstem ; ! NOT-KJ
+A+Comp: áX4b BUSstem ; ! NOT-GG

Handling dialectal variation

When generating sentences or providing the correct answers for the user, we allow only normative forms in the chosen dialect. On the other hand, the live analyser used for the analysis of the user input accepts all correct dialect variants of the same grammatical word. We compile one normative but variation-tolerant transducer for analysing the input, and one strict one for each dialect for sentence generation.

In the source code, forms are marked as missing in certain dialects (the default being that all forms occur in all dialects). Below is an example of dialectal variation in the comparative inflection. The resulting transducers give *stuorát* for the KJ dialect and *stuorit* for the GG one, of the adjective *stuoris* "big".

+A+Comp: iX4b BUSstem ; ! NOT-KJ
+A+Comp: áX4b BUSstem ; ! NOT-GG

Sentence generation in the QA game Vasta

One of the main goals of the programs in OAHPA! is to practice language in natural settings with variation in the tasks. In order to provide variation in programs that involve sentential context we implemented a sentence generator. The sentence generator is used in the morphology in sentential context program (Morfa-C), and for generating questions to the QA drill (Vasta).

```
<q level="2" id="qa_ikte">
<type>PRT</type>
<question>
<text>MAINV go SUBJ ikte</text>
<element id="MAINV">
<grammar tag="VIndiPrtPerson-Number"/>
<sen class="ACTIVITY"/>
</element>
<element id="SUBJ">
<sen class="HUMAN"/>
<grammar pos="N"/>
</element>
</question>
</q>
```

Conclusion

By using a sloppy version of the syntactical analyser for North Sámi, combined with a set of error-detection rules, we have been able to build a flexible CALL resource. The programs are modular, and the modules may be improved by adding more materials -- words, tasks, dialogues, levels, words from textbooks. The CG parser framework was originally chosen as parser framework for Sámi due to its extraordinary results for free-text parsing. The present project has shown that CG is well fit for making pedagogical dialogue systems as well.

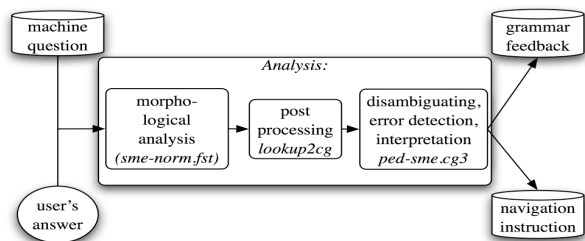
CG-parser in live analysis in the interactive programs Vasta and Sahka

We have chosen not to use multiple-choice, but rather let the student formulate her own answer. To a certain question one may give many kinds of acceptable answers. In Sámi one may change word order, and also add many kinds of particles.

We use a ruleset file which disambiguates the student's input only to a certain extent, because there will probably be grammatical and orthographic errors. The last part of the file consists of rules for giving feedback to the student's grammatical errors, and rules for navigating to the correct next question of in the dialogue, due to the student's answer.

The system question and student answer are analysed together, delimited by the boundary marker ^qst. They first get a morphological analysis (left) , and are then disambiguated, and, if possible, assigned an error tag or a navigation tag.

Schematical view of the process



Buorre beaivi! Humes boastin mu geahčái!
Mun lean áiddo filnen sisa leatn odda orrunsađji. Mus leat lossa viessogáivvut dájpe fasskáris. Hillešit go veahkehit mu?
De mun gillen.
Mus lea TV dáis. Guđe lanjas TV lea du orrunsađis?
Mu TV lea gievvkanis.
Guđe lanjji moai bidje mu TV?
Moai bidje dan hivsega.
The answer should contain an illative.

Grammar feedback

The system may give feedback, as shown here
Here comes an explanation to the example.
&grm-missing-III

```
MAP (&grm-missing-III) TARGET ("guhte") IF  
(1 (N Ill) LINK *1 QDL LINK NOT *1 Ill OR  
DOHKO OR Neg BARRIER S-BOUNDARY);  
<message id="grm-missing-III">The answer  
should contain an illative.</message>
```

```
<Gude>
"guhte" Pron Interr Sg Acc
"guhte" Pron Rel Sg Gen
"guhte" Pron Rel Sg Acc
"guhte" Pron Interr Sg Gen
"latnjii"
"latnja" N Sg Ill
"moai"
"mun" Pron Pers Du1 Nom
"bidje"
"bidjat" V TV Ind Prs Du1
"bidjat" V TV Ind Prt Pl3
"mus"
"mun" Pron Pers Sg1 Gen
"mun" Pron Pers Sg1 Acc
"TV"
"TV" N ACR Sg Acc
"TV" N ACR Sg Nom
"TV" N ACR Sg Gen
"qst"
"asahka" QDL gosa_bidjat_TV
"Moai"
"Mo" N Prop Plc Sg Ill
"mun" Pron Pers Du1 Nom
"Moai" N Prop Plc Sg Ill
"bidje"
"bidjat" V TV Ind Prs Du1
"bidjat" V TV Ind Prt Pl3
"dan"
"dat" Pron Pers Sg3 Acc
"dat" Pron Dem Sg Acc
"dat" Pron Pers Sg3 Gen
"dat" Pron Dem Sg Gen
"hivsega"
"hivisset" N Sg Nom PxSg3
"hivisset" N Sg Loc
"hivisset" N Sg Gen PxSg3
"hivisset" N Sg Acc PxSg3
"<"
" " CLB
```

```
<Gude>
"guhte" Pron Interr Sg Acc &grm-missing-III
"guhte" Pron Interr Sg Gen &grm-missing-III
"latnjii"
"latnja" N Sg Ill
"moai"
"mun" Pron Pers Du1 Nom
"bidje"
"bidjat" V TV Ind Prs Du1
"mus"
"mun" Pron Pers Sg1 Gen
"TV"
"TV" N ACR Sg Acc
"asahka" QDL gosa_bidjat_TV
"Moai"
"mun" Pron Pers Du1 Nom
"bidje"
"bidjat" V TV Ind Prs Du1
"dan"
"dat" Pron Pers Sg3 Acc
"dat" Pron Dem Sg Acc
"dat" Pron Pers Sg3 Gen
"dat" Pron Dem Sg Gen
"hivsega"
"hivisset" N Sg Loc &dia-target
" " CLB
```

Navigation

Navigating inside the dialogue is implemented in CG-rules. The user input is tagged during analysis with information on whether the answer is interpreted as affirmative or negative. In addition, a special tag indicates whether the sentence contains some information that should be stored for the following questions or utterances. The program is thus able to store simple information such as the student's name, place where she lives and for example the type of her car and use this information in tailored utterances.

```
MAP (&dia-target) TARGET NP-HEAD + Ill
IF (*-1 QDL BARRIER S-BOUNDARY LINK **-1 (N Ill)
LINK -1 ('guhte'))(NOT 0 NOTHING);
```

```
<utt type="question" name="gosa_bidjat_TV">
<text>Gude latnjii moai bidje mu TV?</text>
<alt target="hivisset" link="gosa_bidjat_TV">
<text>Dot gal ii heive! Geahčel addasit.</text>
</alt>
<alt target="default" link="gosa_bidjat_beavddi">
<text>Moai guđe dan ovtta dohko.</text>
</alt>
</utt>
```

```
# Picking the age
MAP (&dia-adult) TARGET Num (*-1 QDL LINK @ (Mon_boarís_dan_leat))
@ ('(1-9)[0-9])?);
MAP (&dia-young) TARGET Num (*-1 QDL LINK @ (Mon_boarís_dan_leat))
@ ('(1)[0-9])?);
MAP (&dia-child) TARGET Num (*-1 QDL LINK @ (Mon_boarís_dan_leat))
@ ('(1-9))?);
```

```
<utt type="question" name="How_old_are_you">
<text>Man boarís dan leat?</text>
<alt target="young" link="Do_you_go_to_school_young"/>
<alt target="child" link="Have_you_started_at_school_child"/>
<alt target="adult" link="Do_you_work_adult"/>
<alt target="default" link="Do_you_work_adult"/>
</utt>
```