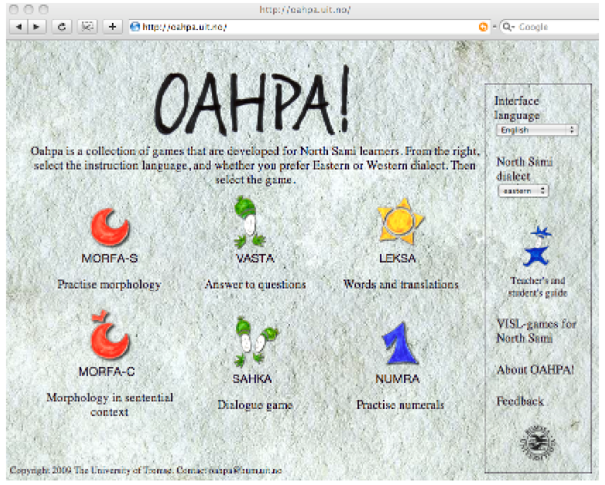
Interactive pedagogical programs based on constraint grammar

(1): Lene Antonsen, Saara Huhmarniemi, Trond Trosterud

http://oahpa.uit.no

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.



Oahpa is a collection of games that are developed for North Sami learners. From the right, select the instruction language, and whether you prefer Eastern or Western dialect. Then select the game.

Practise morphology

Answer to questions

Words and translations

Morphology in sentential context

Dialogue game

Practise numerals

Interface language

North Sámi dialect

Teacher's and student's guide

VISL-games for North Sámi

About OAHPA!

Feedback

Copyright 2009 The University of Tromsø. Contact: oahpa@hum.uit.no

Background

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 pedagogical motivation

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

All 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. In addition, the morphological properties of words are used when making a detailed feedback on morphological errors.

A 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="dl"/>
    <book name="sara"/>
    <book name="algu"/>
  </sources>
</entry>
```

Handling dialectical variation

When generating sentences or providing the correct answers for the user, we wanted to control the selection of word forms to allow only normative forms in the correct dialect. On the other hand, the live analyser used for the analysis of the user input should be tolerant and accept all correct variants of the same grammatical word. Therefore we compiled different analysers/generators for different purposes: one normative but variation-tolerant transducer for analysing the input, and two strict ones for different dialects for sentence generation.

```
+A+Comp:i%>X4b BUStem ; ! NOT-KJ
+A+Comp:á%>X4b BUStem ; ! NOT-GG
```

```
<stem class="bisyllabic" soggi="i">
  <msg case="Ill" number="Sg">i_á</msg>
  <note>láibi > láibái </note>
</stem>

<message id="i_á">Vowel change i > á.
</message>
```

Nord
Lule
Enare
Skolt
Kildin
Ter
Ume
Pite
Sør

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="go_ikte">
  <qtype>PRT</qtype>
  <question>
    <text>MAINV go SUBJ ikte</text>
    <element id="MAINV">
  <grammar tag="V+Ind+Prt+Person-Number"/>
  <sem class="ACTIVITY"/>
    </element>
    <element id="SUBJ">
  <sem class="HUMAN"/>
  <grammar pos="N"/>
    </element>
  </question>
</q>
```

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

Evaluation

The overall evaluation shows that the students answer correctly slightly half of the time. By far the most popular program is the basic morphological drill (but the interactve programs have been logged for a couple of days only).

Program	Correct	Wrong	Total	%
Morfa-S	6920	6323	13243	52.3
Leksa	5659	4248	9907	57.1
Numra	3086	2512	5598	55.1
Morfa-C	1349	1613	2962	45.5
Sahka	322	322	644	50.0
Vasta	19	102	121	15.7
Total	17355	15120	32475	53,44

Evaluating Sahka errors

The 322 logged Sahka errors are distributed along the following lines:

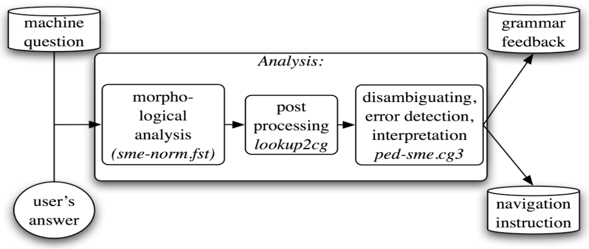
Error type	#	Error type	#
no finite verb	85	wr. case for V-arg	22
orth. error	83	wr. case after Num	10
wrong S-V agr	46	wrong tense	9
no infinite V	30	no postposition	6
wrong V choice	24	wrong word	7

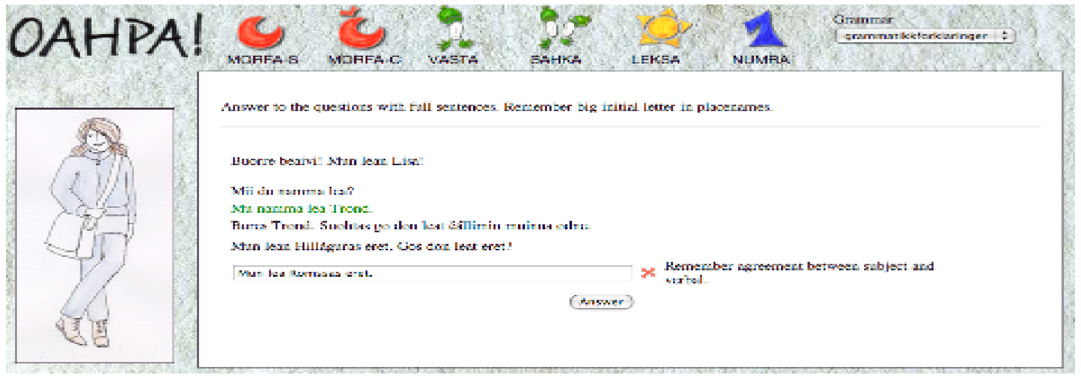
CG-parser in live analysis in the interactive programs Vasta and Sahka

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.

Schematical view of the process


machine question
user's answer
Analysis:
morpho-logical analysis (sme-norm.fst)
post processing lookup2cg
disambiguating, error detection, interpretation ped-sme.cg3
grammar feedback
navigation instruction



Answer to the questions with full sentences. Remember big initial letter in placenames.

Buorre beaivi! Mun lean Lisa!

Mii du namma lea?

Mu namma lea Trond.

Bures Trond. Suohtas go don leat [illegible] muinna odne.

Mun lean Hilláguras eret. Gos don leat eret!

Mun lea Romsas eret.

Remember agreement between subject and verbal.

Grammar feedback

The system may give feedback, as shown above..

```
<utt type="question" name="gosa_bidjat_TV">
  <text>Gude latnjii moai bidje mu TV?</text>
  <alt target="hivsset" link="gosa_bidjat_TV">
  <text>Dat gal ii heive! Geahččal oddasit.</text>
        </alt>
  <alt target="default" link="gosa_bidjat_beavddi">
<text>Moai gudde dan ovttas dohko.</text>
</alt>
</utt>
```

```
<utt type="question" name="Man_boaris_don_leat">
  <text>Man boaris don leat?</text>
  <alt target="young" link="Váccát_go_skuvlla_young"/>
  <alt target="child" link="Leat_go_álgán_skuvlii_child"/>
  <alt target="adult" link="Leat_go_barggus_adult"/>
  <alt target="default" link="Leat_go_barggus_adult"/>
</utt>
```

```
LIST TARGETQUESTION-ACC = ("mii" Acc) ("gii" Acc) ("galle" Acc)
("gallis" N Acc) ;

MAP (&grm-missing-Acc) TARGET TARGETQUESTION-ACC IF (*1 QDL BARRIER
WORK-V LINK NOT *1 Acc OR Neg BARRIER S-BOUNDARY);
```

Analysing the 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 (below), and, if possible, assigned an error tag (here, **&grm-missing-Acc**) (below).

```
"<Maid>"
      "maid" Adv
      "maid" Interj
      "mii" Pron Interr Pl Acc
      "mii" Pron Rel Pl Gen
      "mii" Pron Interr Sg Acc
      "mii" Pron Rel Sg Acc
      "mii" Pron Interr Pl Gen
      "mii" Pron Rel Pl Acc
"<don>"
      "dat" Pron Dem Sg Gen
      "don" Pron Pers Sg2 Nom
      "dat" Pron Dem Sg Acc
"<lohket>"
      "lohkat" V TV Ind Prs Pl3
      "lohkat" V TV Imprt Prs Pl2
      "lohkat" V TV Ind Prt Sg2
"<ikte>"
      "iktit" V TV Ind Prt Pl3
      "iktit" V TV Ind Prs Du1
      "ikte" Adv
"<^qst>"
      "^qst" QDL
"<Ikte>"
      "iktit" V TV Ind Prt Pl3
      "iktit" V TV Ind Prs Du1
      "ikte" Adv
"<mun>"
      "mun" Pron Pers Sg1 Nom
"<lohken>"
      "lohkat" V TV Ind Prt Sg1
"<boares>"
      "boaris" A Attr
      "Boares" N Prop LowercaseErr Attr
"<girji>"
      "girji" N Sg Nom
"<.>"
      "." CLB
```

```
"<Maid>"
     "mii" Pron Interr Pl Acc &grm-missing-Acc
     "mii" Pron Interr Sg Acc &grm-missing-Acc
"<don>"
     "don" Pron Pers Sg2 Nom
"<lohket>"
     "lohkat" V TV Ind Prt Sg2
"<ikte>"
     "ikte" Adv
"<^qst>"
     "^qst" QDL
"<Ikte>"
     "ikte" Adv
"<mun>"
     "mun" Pron Pers Sg1 Nom
"<lohken>"
     "lohkat" V TV Ind Prt Sg1
"<boares>"
     "boaris" A Attr
"<girji>"
     "girji" N Sg Nom
"<.>"
     "." 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.

```
  # Picking the age
MAP (&dia-adult) TARGET Num (*-1 QDL LINK 0 (Man_boaris_don_leat))
    (0 ("([2-9][0-9])"r))  ;
MAP (&dia-young) TARGET Num (*-1 QDL LINK 0 (Man_boaris_don_leat))
    (0 ("([1][0-9])"r)) ;
MAP (&dia-child) TARGET Num (*-1 QDL LINK 0 (Man_boaris_don_leat))
    (0 ("([1-9])"r)) ;
```

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.

References

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI publications in Computational Linguistics. USA.

Eckhard Bick. 2005. PaNoLa: Integrating Constraint Grammar and CALL. applications for Nordic languages. Holmboe, Henrik (ed.): *Nordic Language Technology, Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*. 183–190, København: Museum Tusculanums Forlag.

Eckhard Bick. 2005. Live use of Corpus data and Corpus annotation tools in CALL: Some new developments in VISL. Holmboe, Henrik (ed.). *Nordic Language Technology, Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*, 171–185. København: Museum Tusculanums Forlag.

Johann Gamper and Judith Knapp. 2001. A review of intelligent CALL systems. *Computer Assisted Language Learning* 15(4):329–342.

Trude Heift. 2001. Intelligent Language Tutoring Systems for Grammar Practice. *Zeitschrift für Interkulturellen Fremdsprachenunterricht [Online]* 6(2).

Trude Heift and Devlan Nicholson. 2001. Web Delivery of Adaptive and Interactive Language Tutoring. *International Journal of Artificial Intelligence in Education* 12(4):310–325.

Trude Heift and Mathias Schulze. 2007. *Errors and Intelligence in computer-assisted language learning: parsers and pedagogues*. Routledge studies in computer assisted language learning 2. New York : Routledge.

Fred Karlsson and Atro Voutilainen and Juha Heikkilä and Arto Anttila. 1995. *Constraint grammar: a language-independent system for parsing unrestricted text*. Mouton de Gruyter.

Trond Trosterud. 2007. *Language technology for endangered languages: Sámi as a case study*. http://giellatekno.uit.no/background/lt4ml.pdf University of Tromsø, Norway.

VISL-group. 2008. *Constraint Grammar*. http://beta.visl.sdu.dk/constraint_grammar.html University of Southern Denmark.