Constraint Grammar in Dialogue Systems

Lene Antonsen
University of Tromsø
Norway
lene.antonsen@uit.no

Saara Huhmarniemi
University of Tromsø
Norway
saara.huhmarniemi@helsinki.fi

Trond Trosterud
University of Tromsø
Norway
trond.trosterud@uit.no

Abstract

This article discusses and gives examples of the use of Constraint Grammar as parser engine in parser-based CALL programs for North Sámi. The parser locates grammatical errors in a question-answer program and a dialogue program, and is also used for navigating inside the dialogue.

1 Introduction

The present paper discusses the use of Constraint Grammar (vislcg3) in two different dialogue systems for learning North Sámi: *Vasta* – a QA-drill with open questions, and *Sahka* – a dialogue between program and user within a scenario. The underlying pedagogical goals for both programs are exercising verb inflection, choosing the correct case, and extending the vocabulary of the student.

Constraint Grammar (CG) rules are used for adding tutorial feedback about grammatical errors, navigating in the *Sahka*-dialogue based on the user's answers, and for identifying parts of the user's answer for use in variables later in the dialogue.

Our leading idea was to utilize our existing analyser for Sámi when developing pedagogical programs for language instruction. With vislcg3 we had the possibility of making an intelligent tutoring system with sophisticated error analysis where student tasks could go beyond multiple-choice or string matching algorithms.

Sámi is a language with complex morphology, and it demands much practising before the student reaches necessary skills. However, since Sámi is a minority language, it is common that Sámi students do not receive enough opportunities to practise the language in a natural way. There is also a lack of teaching materials. Therefore, programs accessible on the Internet may be a supplement to the instruction given at school or in universities.

In the following section we describe the basic algorithm for generating questions for *Vasta* and analysing user's input in *Vasta* and *Sahka*. Section 3 shows how CG is used for navigation in the dialogues in *Sahka*, and section 4 shows how tutorial feedback is given with the help of CG rules. In section 5 we present an evaluation of how the system works in real life. The final sections present future perspectives and a conclusion. The programs are available on a web-based learning platform at internet (http://oahpa.uit.no/), which contains six programs (Antonsen, Huhumarniemi and Trosterud, 2009).

2 The system

2.1 Basic grammatical analysis

The basic grammatical analysis of North Sámi is done with finite state transducers (fst) and a constraint grammar parser made at UiT. The relevant resources are the following:

- a morphological fst analyser/generator, compiled with the Xerox compiler *xfst* (Beesley and Karttunen, 2003).
- a morphological disambiguator based on constraint grammar with 3300 manually written rules and a syntactic analyser which adds grammatical function (vislcg3).

The CG parser framework shows extraordinary results for free-text parsing, and Vislcg3 is also used in the VISL-suite of games developed at Syd-Dansk Universitet for teaching grammatical analysis on the Internet (http://visl.sdu.dk/). One of their programs accepts free user input in some of the 7 supported languages. The input is analysed or changed into grammar exercises (Bick, 2005).

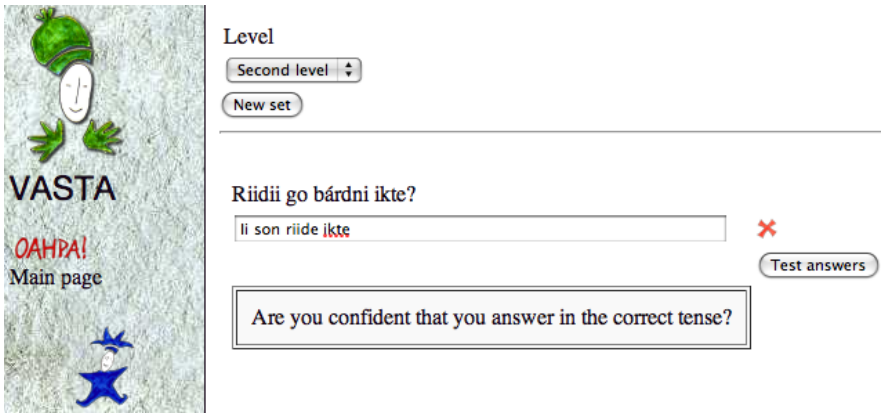

Figure 1: A generated question and a user's answer in *Vasta*. ("Did the boy ride yesterday?" "No, yesterday he does not.")

2.2 Sentence generator

The question-answer drill *Vasta* consists of randomly chosen questions – yes/no-questions and wh-questions. In order to be able to create a large number of potential tasks, we implemented a sentence generator. With the generator we can easily offer variation to the user, instead of tailoring every task with ready-made questions.

A template question matrix contains two types of elements: constants and grammatical units for words selected from the pedagogical lexicon, constrained by semantic sets. The pedagogical lexicon forms a collection of about 2400 words that are considered relevant for the learners of North Sámi in schools and universities. The dialectal variation is taken into account in the lexicon as well as in the morphological generator, and the user may choose eastern or western dialect for the tasks. The sentence generator handles agreement, e.g. between subject and the main verb.

Figure 2 shows a question template in which the main verb (MAINV) is fixed to indicative past tense, but the person and number inflection may vary freely. In Figure 1 on page 2 the same template is realised as a task in *Vasta*. The user's answer triggers a feedback message about the tense of the main verb. Since the content of the MAINV and SUBJ are drawn from the lexicon, the example template may generate around 15 000 different questions.

The question matrices are marked for level, corresponding to the level option chosen by the user, e.g. the basic level 1 has only indicative and no past tense. Because of this we have to fix the inflections in every template to some extent, and there are as many as 111 matrix questions.

```
<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>
```

Figure 2: A question template (MAINV question-particle SUBJ yesterday).

2.3 The analysing process

Both the question and the answer are analysed with the morphological analyser and then the result is postprocessed to cg3-format and passed to the CG3 rule component (cf. Figure 3). The question and user's answer pairs are merged, and analysed as one text string. The question mark in the question is exchanged for a special symbol ("^qst" or "^sahka" QDL), as shown in the analysed question-answer pair in Figure 5 on page 4. We use these symbols, rather than the question mark itself, in order not to introduce a sentence delimiter in the analysis, since we want to refer to the question and the answer separately in the rules (left or right side of the QDL), but also treat

the question-answer part as one unit. Many of the constraints are based upon the grammar and semantics of the question – e.g. the tense and person inflection of the verb, the case of NP in the answer and so on. The question itself restricts the possible interpretation of the input.

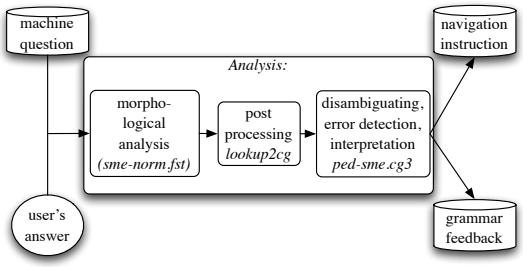

Figure 3: Schematical view of the process.

The vislcg3-rule set consists of two parts. The first part is a rule set, which disambiguates the user's input only to a certain extent. The rule set is relaxed compared to the ordinary disambiguator, in order to be able to detect relevant readings despite of a certain degree of grammatical and orthographic errors in the input. The second part of the rule set contains rules for giving feedback to grammatical errors, and rules for navigating to the next question or utterance in the dialogue, based on the user's answer. In this paper, we concentrate on the rules for giving feedback for the user and navigating in the dialogue.

3 Navigating in the dialogues

In the *Sahka* dialogues the main goal has been to create a feeling of a natural dialogue. One of the ways to achieve that goal is reacting to the user's input. When the input is morphologically analyzed, the CG rules are used for assigning tags to the question-answer pairs, which are then used for selecting appropriate questions and navigating in the dialogue. The dialogues deal with different topics. The "first meeting dialogue", for example, treats topics such as age, family, working place/school, car and so on. Navigation between the topics is achieved by recognizing and tagging the content of the user's answer in CG rules and providing the analysis to the *Sahka*-engine. In addition, it is possible to assign a target tag to certain information types; the system may e.g. collect name, car brand and so on, and use it as a variable in the follow-up questions.

The CG rules used in the dialogue processing may be divided into two types: general rules that may target any question-answer pair and question-specific rules that are tailored for a specific question.

3.1 Rules for specific questions

Since the functionality of *Sahka* is more dependent upon correct analysis of the content of user's answer, the questions in the dialogues do not vary freely as in *Vasta*. Every question is a text string and has its own unique name assigned to the QDL. This enables writing question specific CG rules and accessing the question from other questions.

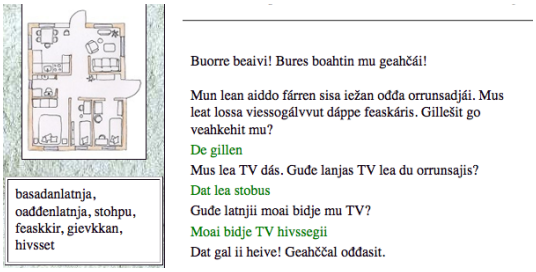

Figure 4: From *Sahka*. ("In which room should we place the TV?" "We should place it in the toilet." "That is not a good idea. Try again.")

Consider an example dialogue from *Sahka*. In Figure 4 the setting is a visit to a friend who has moved into a new flat, and needs a helping hand with moving the furniture. We have come to the third question and the next question in the dialogue is selected depending on the answer. In Figure 5 the analysis assigns two navigation tags to the question-answer pair. The rule for assigning the tag *&dia-hivsset* is shown in Ex. (1), the other one is explained in section 3.2.

(1) MAP (&dia-hivsset) TARGET QDL IF
(0 (where_place_TV))
(*1 ("hivsset") BARRIER Neg OR ROOMS) ;

This special rule for the question with the identifier *where_place_TV* adds the tag *&dia-hivsset* to the QDL in the question-answer pair if the answer contains the word *hivsset* (toilet). The barrier prevents the rule from working if the negation verb or a word from the set denoting rooms intervenes between the QDL and the word *hivsset*. The barrier will prevent assigning the tag to answers, which negate the possibility of putting the TV to the toilet, or giving the toilet as only

one of more possibilities.

```
"<Gude>"
   "guhte" Pron Interr Sg Gen
"<latnjii>"
   "latnja" N Sg Ill
"<moai>"
   "mun" Pron Pers Du1 Nom
"<bidje>"
   "bidjat" V TV Ind Prs Du1
"<mu>"
   "mun" Pron Pers Sg1 Gen
"<TV>"
   "TV" N ACR Sg Acc
"<^sahka>"
   "^sahka" QDL where_place_TV &dia-hivsset
"<Moai>"
   "mun" Pron Pers Du1 Nom
"<bidje>"
   "bidjat" V TV Ind Prs Du1
"<TV>"
   "TV" N ACR Sg Gen
"<hivssegii>"
   "hivsset" N Sg Ill &dia-target
"<.>"
   "." CLB
```

Figure 5: Assignment of navigation tags is done together with the disambiguation.

```
<utt type="question" name="where_place_TV">
  <text>Gude latnjii moai bidje mu TV?</text>
  <alt target="hivsset" link="where_place_TV">
  <text>Dat gal ii heive! Geahčča1 oddasit.</text>
       </alt>
  <alt target="default" link="where_place_table">
<text>Moai gudde dan ovttas dohko.</text>
</alt>
</utt>
```

Figure 6: From the a dialogue file. (”In which room should we place the TV?” Alt. ’toilet’: ”That is not a good idea. Try again.” Default: ”We carry it there together.”)

When the *Sahka*-engine reads the CG-output, it recognizes the dia-tag and searches for a next instruction based on the tag. Every question contains links to alternative questions that are selected based on the recognized tag. In addition, there is a default link in case a navigation tag was not present in the CG-input. In Figure 6 there are two alternative links for the answers to the question in Figure 5. One of them is connected to the *&dia-hivsset* tag and will give the answer ”That is not a good idea. Try again.” The other link is default and leads to the next question in the dialogue.

Another example of a question specific dialogue navigation rule comes from yes/no-questions where the user often provides more information than what was asked for. E.g. to the question ’Do you have children?’, the user can answer ’Yes, I have two children.’ In the dialogue, the next question would normally be ’How many children do you have?’. To avoid this question when the information was already provided, we have a pass-tag for omitting the next question. In this case, the pass-tag is added to the question with identifier *do_you_have_children* if the answer contains a numeral, as shown in example (2):

(2) MAP (&dia-pass) TARGET QDL
(0 (do_you_have_children) LINK *1 Num) ;

Let us consider a couple of examples showing how the dialogue may be branched to different questions and topics. In Figure 7 there are different follow-up questions for the answer to “What kind of car do you have?” If the car brand is in the lexicon, the system picks up the car type and uses it in a variable in the next question, e.g. ”Is Ford a good car?”, and if it is not in the lexicon (it can e.g. be a spelling error or a joke from the user), the next question will be “Is it a car?”. There is also an alternative link for a negative answer (“Do you want to buy my car?”), and the default leads to a comment, which closes this topic.

```
<utt type="question" name="What_kind_of_car">
  <text>Makkár biila dus lea?</text>
  <alt target="target" variable="CAR" link="Is_it_a_good_car"/>
  <alt target="unknown" variable="CAR" link="Is_it_a_car"/>
  <alt target="neg" link="Do_you_want_to_buy_my_car"/>
  <alt target="default" link="car_closing"/>
</utt>

<utt type="question" name="Is_it_a_good_car">
  <text>Lea go CAR buorre biila?</text>
  <element id="CAR">
  </element>
```

Figure 7: Alternative links due to the answer of ’What kind of car do you have?’.

```
  <utt type="question" name="How_old_are_you">
    <text>Man boaris don leat?</text>
    <alt target="young" link="at_school_young"/>
    <alt target="child" link="begin_at_school_child"/>
    <alt target="adult" link="job_adult"/>
    <alt target="default" link="job_adult"/>
  </utt>
</topic>
```

Figure 8: Alternative branches due to the age of the user. The question is ’How old are you?’.

Whenever a topic is closed, the dialogue proceeds to the next topic. For example, an answer from the user about her age will induce a tag which is used for navigating to different branches of the dialogue based on the age of the user, as in Figure 8. The tag for age is assigned with a regular ex-

pression inside a CG rule, as in the examples (3), (4) and (5):

(3) MAP (&dia-adult) TARGET Num
(*-1 QDL LINK 0 (How_old_are_you))
(0 (”([2-9][0-9])”r)) ;

(4) MAP (&dia-young) TARGET Num
(*-1 QDL LINK 0 (How_old_are_you))
(0 (”([1][0-9])”r)) ;

(5) MAP (&dia-child) TARGET Num
(*-1 QDL LINK 0 (How_old_are_you))
(0 (”([0-9])”r)) ;

Users in the age-group below 20 proceed to a topic about going to school, while the older users are asked about their work. The question contains a default link as well, since some users have fun telling they are 1000 years old.

3.2 General rules

Most of the cg3-rules are general rules that apply to all question-answer pairs. Consider example (7), which generalizes over the question marker set in (6):

(6) LIST TARGETQUESTION-ACC = (”mii” Acc)
(”gii” Acc)(”galle” Acc) (”gallis” Acc) ;

(7) MAP (&dia-target) TARGET NP-HEAD + Acc IF
(*-1 QDL BARRIER S-BOUNDARY LINK *-1
TARGETQUESTION-ACC LINK NOT 0 Num)
(NEGATE *1 (N Acc) BARRIER VERB OR
CC)(NOT 0 NOTHING) ;

This is a general target rule for questions, which requires an answer in the accusative. S-BOUNDARY is a set of words and tokens which marks the end of the (sub)sentence. NOTHING is a set of indefinite pronouns like ”nothing” and ”nobody”. There are similar rules for other cases.

There are also general rules for tags marking whether the answer is interpreted as affirmative or negative, as in Ex. (8):

(8) MAP (&dia-pos) TARGET QDL IF
(*-1 Qst OR go)(NOT *1 Neg);
MAP (&dia-neg) TARGET QDL IF
(*1 Neg BARRIER S-BOUNDARY);

In Sámi a yes-no question is indicated by a question particle ”go”, which can be a separate word or cliticized to the word to the left, which then gets a *Qst* tag in the analysis.

3.3 Storing information

It is useful to store some information about the user during the dialogue, such as name and age of the user. This information may be used in questions later, and give an impression of familiarity. These are implemented using special tags, such as in the examples (9) and (10):

(9) If the name is not in the lexicon:
MAP (&dia-target) TARGET QMRK IF
(*-1 QDL BARRIER (&dia-target) LINK 0
(What_is_your_name)) ;

(10) The name is in the lexicon:
MAP (&dia-target) TARGET Prop IF (*-1 QDL
BARRIER (&dia-target) LINK 0
(What_is_your_name)) ;

The set QMRK contains the question mark, and is given if the name is not in the lexicon, which is quite common with names. Both rules have &dia-target as barrier so it will hit only the first name, if there are many. There are similar rules and tags for information concerning place names, car brands and so on, and the information is used by the system in variables in tailored questions or utterances.

4 Tutorial feedback

The system gives tutorial feedback about grammar errors both in *Vasta* and *Sahka*. The feedback is generated from the grammar error tags, which are assigned during the disambiguation analysis. It should be noted that the system uses the grammatical analyser on the fly, exploiting full lexicons. This allows the user's answer to contain any Sámi word, also words that are not restricted to the pedagogical lexicon.

4.1 Grammar errors

```
"<Guđe>"
   "guhte" Pron Interr Sg Gen &grm-missing-Ill
"<latnjii>"
   "latnja" N Sg Ill
"<moai>"
   "mun" Pron Pers Du1 Nom
"<bidje>"
   "bidjat" V TV Ind Prs Du1
"<mu>"
   "mun" Pron Pers Sg1 Gen
"<TV>"
   "TV" N ACR Sg Acc
"<^sahka>"
   "^sahka" QDL where_place_TV
"<Moai>"
   "mun" Pron Pers Du1 Nom
"<bidje>"
   "bidjat" V TV Ind Prs Du1
"<TV>"
   "TV" N ACR Sg Gen
"<gievkkanis>"
   "gievkkan" N Sg Loc
"<.>"
   "." CLB
```

Figure 9: A grammar error tag is assigned.

In the question in Figure 9, the systems asks ”In which room should we place the TV?” The

user answers ”Moai bidje TV gievkkanis” (’We should place the TV in the kitchen’), with locative “gievkkanis” rather than the correct illative “gievkkanii”. The CG parser disambiguates the input, and the sentence matches the structural description of the general CG rule in example (11):

(11) MAP (&grm-missing-Ill) TARGET (”guhte”) IF (1 (N Ill) LINK *1 QDL LINK NOT *1 Ill OR ADV-ILL OR Neg BARRIER S-BOUNDARY) ;

The rule adds a grammar-error-tag *&grm-missing-Ill* to the sentence analysis triggered by the interrogative pronoun followed by a noun in illative. This combination requires an illative form in the answer, when there is no illative form nor adverb with illative interpretation nor negation verb in the answer. The *Sahka*-engine generates a tutorial message based on the error-tag, given in example (12):

(12) <message id=”grm-missing-Ill”>The answer should contain an illative. </message>

One of the pedagogical goals behind the programs is that the user should practice inflecting the finite verb correctly. A central requirement is thus that the user answers with full sentences containing a finite verb. To encourage the user to practice also difficult verbs, she has to use the same verb as in the question. The CG rule in example (14) controls the choice of verb for the answer, and it uses a regular expression-based tag (a so-called *sticky tag*). The verb is identified via a regular expression $.*$ (cf. (13)), and the rule in (14) is triggered if it does not find the same verb lemma in both the question and the answer.

(13) LIST VERBLEMMA = (”.*”r) ;

(14) MAP (&sem-answer-with-same-verb) TARGET FINVERB (NOT 0 Neg OR AUX-SET) (0 $$VERBLEMMA LINK *-1 QDL BARRIER S-BOUNDARY OR FINVERB LINK NOT 0 EXEPTION-QUESTIONS LINK *-1 FINVERB -1 BOS LINK NOT 1 $$VERBLEMMA)) ;

BOS is the left border of the sentence. Pro-verbs get a special treatment, and a question containing a pro-verb will accept any verb in the answer. There are also exceptional rules for some auxiliary verbs and for some questions, like for the question ”What is your name?”, which will more naturally be answered without a verb.

In *Vasta* the pronouns are not allowed to be interpreted inclusively (*we / you*, not *we / we*), but in *Sahka* they follow the logic of the scenario. This is the main reason for why *Sahka* has a slightly different rule set compared to *Vasta*. To indicate the type of the program in the morphological analysis, the delimiter between question and answer in *Sahka* is ”^sahka” instead of the delimiter tag ”^qst” used in *Vasta*.

Some of the questions in the *Sahka* dialogues are made for special grammatical training such as adjectival comparison. These questions populate a whole section of rules in the CG file. The rules add specific feedback to the potential errors.

The user will get only one feedback at a time, so the error tags are ordered partly as natural progress for error correction, and partly according to the likeliness of the error. First of all, the user will get feedback about spelling errors. If there is no agreement between subject and verb, then she will get feedback on the verb form, and not on the pronoun, given the assumption that the error is in the verb form rather than in the pronoun.

Grammar errors we have rules for, include

- verbs: finite, infinite, negative form, correct person/tense according to the question
- case of argument based upon the interrogative
- case of argument based upon valence
- locative vs. illative based upon movement
- subject/verbal agreement
- agreement inside NP
- numeral expressions: case and number
- PP: case of noun, pp based upon the interrogative
- time expressions
- some special adverbs
- particles according to word order
- comparison of adjectives

4.2 Misspellings

The user’s misspellings form the largest distinct problem for the functionality of the game. If the spelling error gives rise to a non-existing word form, then the message to the user is ”The word form is not in our lexicon, can it be a spelling error?”, which often is not of enough help to the user. A human reader would be able to read the answer in a robust way, and detect what the user intended to write. Simulating this ability is not an easy task.

Running the feedback through an ordinary speller engine is not a good solution, since the speller will come up with a large number of suggestions, without being able to choose between

them. A possible solution would be to run a morphological analysis on the speller suggestions, and let a CG component pick the most likely candidates. The problem is that the current North Sámi speller (`http://divvun.no`) is made for native speakers and corrects mainly typing errors. In *Vasta* and *Sahka*, we would need a correction mechanism for errors due to wrong choice of affixes.

As a partial solution, we have added rules to the morphophonological rule file for typical spelling errors in e.g. place names. This enables the system to give a specific feedback in case of typical misspellings of place names. If the place name still is not recognized by the analyser, the feedback in the dialogue is "I haven't heard about X. Is it a place?", and the navigation proceeds to the next question.

The misspelling can also give rise to another word form of the same lemma. For such cases we have made rules based on the sentential context. The challenge is to give a feedback according to what the user thinks she has written, because she is probably not aware of the unintended word form. E.g. if the consonant gradation is incorrect in an attempted singular locative, the word form will be a nominative with possessive suffix Sg3. The learner will probably not know the possessive suffixes yet, so referring to it would not be useful. Instead, she gets the feedback: "Do you mean locative? Remember consonant gradation."

A more difficult problem emerges when the spelling error gives rise to an unintended lemma. Then the challenge is again to give feedback according to what the user thinks she has written. The feedback has to be tailored to what we know about the user's interlingua – and we have made some rules for sets of typical unintended lemmas. Some of them are systematic, such as the Sg2 of a verb incorrectly used after the negative verb, will result in a ConNeg form of a derived verb.

4.3 Metacomments

The *Sahka* program is intended to mimic a natural dialogue. But there are some restrictions in the possible input from the user; the system has to be able to analyse the input, and the answers should be pedagogically meaningful for the user. To remind the user of that, the system sometimes give metacomments to the user, like the following:

- "Answering *I-don't-know* is too simple. Try again."
- "Your answer must always contain a finite verb."
- "You must use one of the words in the wordlist in the left margin."
- "You have not used the correct adjective. Try again."

5 Evaluation

The evaluation of *Sahka* and *Vasta* was done when the programs had been available on internet for three months. The user's input and the feedback from the system were logged for the last two weeks of the period. The log shows that *Sahka* has six times as many queries as *Vasta*, so users clearly prefer the former one.

The system gave 156 tutorial feedbacks for the two programs during the two weeks. Breaking down the precision numbers on type of feedback, we got the picture shown in Table 1. Of 27 erroneous judgements, 16 were due to technical malfunction, 9 to wrong syntactical and 2 to wrong lexical analysis.

Rule type	**corr.**	**wrong**	**corr. %**
wrong tense	7	0	100,0
wr. V after neg	3	0	100,0
no infinite V	1	0	100,0
orth. error	44	2	95,7
wr. case V-arg	26	4	86,7
no finite verb	19	4	82,6
wr. S-V agreem.	17	8	68,0
wrong V choice	7	4	63,6
wrong word	4	4	50,0
wr. case after Num	1	1	50,0

Table 1: Feedback precision for different rule types.

As shown in Table 1 not all of the rule types mentioned in 4.1 have been in use during this period. These rule types have not been used:

- agreement inside NP (except for numeral expressions)
- nominal case inside PP
- time expressions
- word order errors for particles

The reason is probably that the users do not write more complex language than they have too. E.g. they don't answer with a complex NP if they

can answer with just a pronoun or a noun, they don't write a time-expression with PP if the can answer with an adverb instead, and they don't use optional particles if they are unsure of where to put them. The price we pay for the free input strategy is that the users are not forced to exercise more complex language.

Table 2 on page 9 shows different kinds of error types the system has identified in the user's sentences, these we call *positives*. If it really is an error, then we call it it a *true positive*, if not, then it is a *false positive*. A sentence not flagged as an error by the system is counted as a *negative*, and we distinguish between *true negatives* (correct answers) and *false negatives* (erroneous answers which the system did not detect).

We measured *precision* (correctly identified errors/all diagnosed errors), *recall* (correctly identified errors/all errors), and *accuracy* (correct judgements/cases). For the error types we target, precision = 0.85, recall = 0.93, and accuracy = 0.89 (N=277). Better recall than precision indicates that very few errors slip through, at the price of erroneously identifying some correct forms as errors. The system is thus a bit too critical towards the students: It almost never lets through a (targeted) mistake. In this pedagogical setting, a goal for future work is improving precision (avoiding erroneous error flagging), perhaps even with the risk of a lower recall.

6 Future perspectives

We have started the work with improving the system. Among our future plans are:

Implementing a speller. Because the misspellings are the biggest problem for the users, we will implement a speller. We will give relevant suggestions to the user by analysing the list of suggestions according to the context with CG, and also implement weighted lexical transducers, see (Linden and Pirinen, 2009). For the weighting we will use the pedagogical lexicon and the North Sámi corpus as a training corpus.

Implementing a topic option in *Vasta*. Today *Vasta* generates questions randomly within each level based on grammar difficulty. The log shows that this program is not as popular as the *Sahka*. We are planning to make it more interesting for the users by restricting the semantic sets for the variables in the question templates according to topics, and give the user's a topic option as well.

Sentence building from a fixed set of lemmas. We are also considering forcing the user to construct more complex phrases and also use more particles, by deciding what lemmas the user should use, as a supplement to the other programs. Available on internet is *e-tutor* – a program for teaching German to foreigners (Heift, 2001), at `http://e-tutor.org/`. e-tutor gives very good feedback to student's errors, but the possible input is restricted to a set of lemmas by means of which she has to construct a sentence. In this way the user is forced to write more complex phrases. Figure 10 shows an example from the program.

Conduct studies on Oahpa in actual use. Investigating how Oahpa works in actual use in the classroom will be important in the work with improving the system.

Porting the programs to more Sámi languages. For Lule Sámi a morphological analyser is available, and we have started making a CG disambiaguator. For South Sámi a morphological analyser will be finished in 2010.

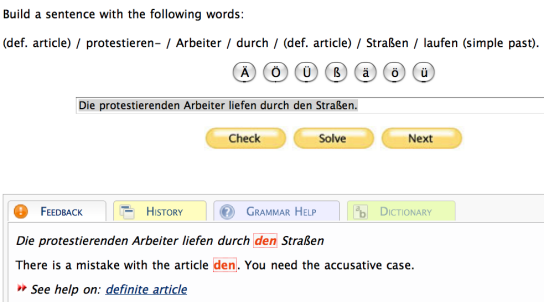

Figure 10: An alternative to free input is e-tutor.

7 Conclusion

The paper has shown how we use vislcg3 for pedagogical dialogue systems for North Sámi. Vislcg3 is used in many ways: By relaxing the analysis of the input string, we are able to find errors made by the user, and assign feedback tags to the analysis. Secondly, by analysing the semantics of the user's input, and assigning semantic tags to the input, we are able to navigate through the dialogue according to user feedback. And finally, we can assign

Error type	true pos.	false pos.	true neg.	false neg.	precision	recall	accuracy	F-ms.
Gramm. error	641	234	769	7	0,73	0,99	0,85	0,84
Semant. error	805	69	764	12	0,92	0,99	0,95	0,95
Orthogr. error	875	0	776	0	1	1	1	1
Other error	695	180	751	25	0,79	0,97	0,88	0,87
	3016	483	3060	44	0,86	0,98	0,92	0,92

Table 2: Precision, recall and accuracy for different error types.

tag to information in the user's input and use it in the program's questions or utterances.

The CG formalism has a great potential for use in pedagogical settings. It is robust enough to handle erroneous data, and at the same time flexible enough to give both general corrections, and corrections targeted at specific words in specific settings.

We have seen that a major problem is spelling errors. Whether CG is able to offer a solution for this problem as well, remains a topic for future research.

Acknowledgments

Thanks to the faculty of Humanities at the University of Tromsø, and the Sámi Parliament in Norway, for funding the project.

References

Lene Antonsen, Saara Huhumarniemi and Trond Trosterud. 2009. Interactive pedagogical programs based on constraint grammar. *Proceedings of the 17th Nordic Conference of Computational Linguistics*. Nealt Proceedings Series 4.

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

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.

Krister Lindén and Tommi Pirinen. 2009. Weighted Finite-State Morphological Analysis of Finnish Compounding with HFST-LEXC. *Proceedings of the 17th Nordic Conference of Computational Linguistics.*. Nealt Proceedings Series 4.

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

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

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