COLING 82, J. Horecký (ed.)
North-Holland Publishing Company

AN EXPERIMENTAL PARSER

Anna Sågvall Hein

Center for Computational Linguistics
University of Uppsala
Sweden

Uppsala Chart Processor is a linguistic processor for phonological, morphological, and syntactic analysis. It incorporates the basic mechanisms of the General Syntactic Processor. Linguistic rules as well as dictionary articles are presented to the processor in a procedural formalism, the UCP-formalism. The control of the processing rests upon the grammar and the dictionaries. Special attention has been devoted to problems concerning the interplay between different kinds of linguistic processes.

INTRODUCTION

Uppsala Chart Processor (UCP) includes the basic mechanisms of a refined version of the general syntactic processor (GSP), presented by M Kay in 1977, see /1/. An integrated part of the UCP is the UCP grammar formalism. The starting point for the UCP formalism was the 'reversible grammar formalism', see /2/.

The reversible grammar formalism includes the specification of a format for linguistic descriptions, - the structure - , and a set of reversible - apt for generation as well as analysis - grammar operators.

In the UCP we keep the idea of a procedural grammar formalism, consisting of a set of grammar operators, and also the structure for representing linguistic descriptions. We don't, however, impose any requirements of 'reversibility' upon the operators. UCP in its present version is designed for analysis.

Our extensions and modifications were motivated by our attempt to apply the conceptual and computational machinery of the GSP and the reversible grammar formalism to phonological and morphological analysis of Finnish and to morphological and syntactic analysis of Swedish.

Basically, the extensions consist in creating the means for embedding various kinds of linguistic processing - dictionary search, rewriting, phonological, morphological, and syntactic analysis - in the framework offered by the GSP and for arranging for some interaction between them. Our solution to these problems is based upon the procedural nature of the UCP formalism. We created the means that were needed by defining new grammar operators.

All language specific information of a UCP parser is expressed in the procedural UCP formalism. This is true, not only for rules but also for dictionary articles. An optional part of a UCP parser is a 'filter grammar', stating what rules should apply to what linguistic units. The filter grammar itself is expressed in the UCP formalism.

INTRODUCING NON-REVERSIBLE GRAMMAR OPERATORS

In the reversible grammar formalism testing as well as the assignment of features is carried out as unification operations. Using the unification operation for feature testing results in an assignment as soon as one of the features is missing. In order to avoid such effects, one has to test for the presence of features before the unification operation is applied. This turned out to be a serious drawback in our Finnish application where an abundance of feature tests have to be carried out to guarantee compatibility between the morphs. In order to overcome this problem we defined an additional identity operator.

Another characteristic feature of the reversible grammar formalism is the lack of an or-operator (dependent disjunction). The only operator available for handling alternatives is an operator that initiates parallel processes, one for each alternative (independent disjunction). This had a serious effect on processing efficiency. The situation was substantially improved by the dependent disjunction operator, that we defined for the UCP formalism.

LOOKING AHEAD IN THE UCP

The GSP doesn't provide any means for looking ahead. We felt a need for such a possibility in formulating our Finnish rules. For instance, the Finnish string 'maa' represents a full word (in the nominative singular) if followed by a separator, a plural stem (if followed by 't'), and a singular stem in the remaining cases. Now, unless we have a way of inspecting the following character we will have to store these three alternatives and also initiate further processing for each case respectively. - There are numerous analogues cases in the recognition of syllables.

The problem was solved in the following way. In the reversible grammar formalism there is only one operator available for storing an analysis - introducing an inactive edge into the chart. This operator (DONE, in our notation STORE) has the effect of inserting an edge spanning the sequence of vertices covered by the inactive and active edge, involved in the task at hand. We defined an additional operator (MINORSTORE) which adds a new inactive edge to the chart spanning the same sequence of vertices as the active edge only. Disposing of this operator, we may postpone the storing of an analysis until the following edge (character, word, phrase) has been inspected.

RULE INTERPRETATION AND REWRITING

Characteristic for Finnish is the abundance of phonological and morphophonemic alternation. For the handling of these problems we decided to formulate a new grammar operator that would allow us to integrate rewriting operations with the interpretation of phonological and/or morphological rules. For instance, rewritings which occur in the domain of the syllable, such as e g three vowel simplification should favourably be integrated with the recognition of the syllables in the phonological processing. A generalization of the STORE operator provided us with this facility. STORE without an argument has the effect outlined above. However, in our generalized version STORE may take an arbitrary number of arguments, each argument corresponding to a set of linguistic features. The interpretation of such an 'extended' store operation leads to the introduction of a sequence of (inactive) edges into the chart, labeled with the linguistic descriptions of the store arguments.

RULE INTERPRETATION AND DICTIONARY SEARCH

The strategy that we wanted to adopt for the morphological analysis of Finnish involved an interplay between dictionary search and rule interpretation.

The states in the morphological grammar were to correspond to two kinds of rules, i e 'acceptance rules' and 'morphotactic rules'. By an acceptance rule we understand a rule, where a tentatively recognized morph, typically the stem, is being inspected, and as a result of this inspection accepted or rejected for further processing. A reason for rejecting a stem candidate may be, that the string which has been found to match a stem entry in the dictionary has been rewritten in a fashion that conflicts with the restrictions for alternation stated about that stem. Apart from these tests, an acceptance rule should also contain a prediction about the following morph. We wanted to express this prediction as a call to start searching for a certain kind of morph in the appropriate suffix dictionary (plural suffix dictionary, case suffix dictionary etc). By a morphotactic rule we understand a rule where the compatibility between adjacent morphs is being verified. In addition, these rules should also give the appropriate predictions in the same manner as the acceptance rules. The entire morphological processing would start by searching for a stem in the main dictionary, and the appropriate grammar rules would be invoked from the dictionary articles. For further details, see /3/.

The strategy outlined above presupposes the existence of means for initiating dictionary search from the grammar and invoking grammar rules from the dictionary articles.

We solved the first problem by defining a new operator PROCESS (dic.name) with the effect of initiating dictionary search in the dictionary named dic.name.

Our first solution to the second problem was the following. Apart from the linguistic description of the lexical item, a dictionary article would include two reserved fields for the names of the grammar rules to be invoked, one for the acceptance rules and one for the morphotactic rules.

However, in turning our attention to grammatical analysis of Swedish, we got aware that in order to experiment freely with different kinds of interaction between the grammar and the dictionaries, we needed more flexibility.

Not only would we like to be able to initiate dictionary search from the grammar, but also sometimes from a dictionary article.

Furthermore, it was alright to be able to invoke a grammar rule from a dictionary article, but it was not sufficient. Sometimes there were alternatives; sometimes an alternative could be ruled out by a simple look-ahead. As a matter of fact, we realized that what we actually wanted in formulating the dictionary articles was the richness of the grammar formalism for experimenting with the different actions that might be taken at the retrieval of lexical items. In this approach, when a lexical entry has been retrieved the fragment of 'grammar' associated with it is being interpreted in exactly the same way as a grammar rule.

Hereby, we eliminate the formal differences between grammar rules and dictionary articles. What theoretical implications this approach may have is not clear to us yet. - On the implementational level this, of course, streamlines the processor.

In the Finnish application, when a tentative stem had been found in the dictionary, a grammar rule was invoked with the effect of inspecting the stem, and of giving a prediction of the following morph. This was handled by a piece of LISP code which inserted an incomplete (active) edge from and to the last vertex of the stem, directed from right to left.

In a syntactic application we want to have the possibility of invoking grammar rules from signals in the text, for instance a preposition phrase rule at the recognition of a preposition. This situation is analogous to the Finnish stem case. However, in designing the operator that we wanted here, we chose to have it working from left to right. Consequently, it was defined in such a way that it

would insert an incomplete edge from and to the initial vertex of the key word (e g Finnish stem or preposition), directed from left to right. The operator that we talk about is MAJORPROCESS (rule.name). We may say that it fulfils a function of looking ahead, since it gives us the possibility of invoking a rule first when we are sure that at least the first transition will succeed.

For an illustration of the UCP formalism we present the dictionary article for the Swedish preposition 'med' (Engl. 'with')

```
med  <& lex> :=: 'med,
     <& morph.cat> :=: 'word.cand,
     <& syn.cat> :=: 'prep,
     advance,
     <* char :type> = 'sep,
     minorstore,
     majorprocess (indeclineable),
     majorprocess (prep.phrase),
     process (sep);
```

Figure 1

The 'rule' is a conjunction (',') of 9 operations. If one of the operations fails, the whole conjunction fails. In the first three operations features of 'med' are assigned. Then there is an advancement in the string for an inspection of the following character. If it turns out to be a separator (blank or sign of punctuation) the result, i e a complete edge with a linguistic description as specified by the assignment operations, is stored by means of MINORSTORE. After that, the INDECLINABLE grammar rule is invoked, which builds words from word candidates followed by spaces. There is also a call to start processing the PREP.PHRASE rule, which subsequently will recognize the preposition and look for a following NP. The last operation starts search in the separator dictionary.

The whole rule, apart from the first assignment, will apply to most prepositions. Instead of having to repeat the whole rule for every preposition in the dictionary, we may collect the common part under its own rule name in the grammar, and have this subrule interpreted at the retrieval of such a preposition by simply giving its name in the dictionary article.

Using this facility, we may formulate the dictionary article presented in figure 1 simply as shown in figure 2.

```
med  <& lex> :=: 'med,
     preposition.action;
```

Figure 2

As mentioned above, there is a rule in the grammar for indeclineable words. It recognizes an indeclineable word as consisting of a word candidate followed by a space sign. In other words, the space is considered as a morphological constituent. Signs of punctuation, however, are regarded as syntactic constituents. According to the orthographic convention, a sign of punctuation is attached to the word candidate without a preceding space sign. Here it fulfils two functions, i e denoting the end of the word and the end of a syntactic constituent. We solve this problem by making the space sign explicit by means of a rewriting operation in the dictionary articles of the signs of punctuation. In figure 3 we present the dictionary article of the comma sign for an illustration.

```
<& first morph.cat> :=: 'space,
<& second syn.cat> :=: 'sign.of.punct,
<& second lex> :=: 'comma,
store (<& first>,<& second>);
```

Figure 3

PARALLEL PROCESSING LEVELS

Morphological analysis of Finnish requires the support of phonological analysis. For instance, in the illative singular we need phonological information about the stem, such as number of syllables and properties of the last syllable, in order to predict the correct case allomorph.

Whereas the morph is the fundamental segment in the morphotactic processing, the syllable is the fundamental segment in the phonological processing. Consequently, they should be considered as different linguistic processes. If we want to parse a Finnish word in one pass, these two processes have to be carried out in parallel. Furthermore, there must be a means for communicating the results of the two processes between them.

We hypothesize that there is a superordinate 'word recognition grammar' for Finnish, stating the relationship between phonological and morphological processing in the following way

```
WORD:    PROCESS(phon.rules), PROCESS(morph.rules);
```

WORD is the name of a state (the single one) in the word recognition grammar. It contains a conjunction of two operations, the order between the operations being immaterial. The interpretation of this rule has the effect that the first rule of the phonological grammar, phon.rules, is invoked as well as the first rule of the morphological grammar, morph.rules. Hereby, phonological and morphological analyses are being carried out in a pseudo-parallel manner, independently of each other. - It should be noted, that the PROCESS operator mentioned above which was originally defined for initiating dictionary search has been generalized to account for the invocation of grammars and grammar rules as well. What is a dictionary name and what is the name of a grammar or a grammar rule is part of the self-knowledge of the UCP.

The only medium of communication between the two processes is the chart, being the 'common note book' of the entire processing, in which all results are stored. In order for the morphological processing to be able to use the results of the phonological processing - e g when a stem has been found - we must provide a means for accessing this information (number of syllables etc).

Lexical information about the stem, resulting from the dictionary search process (initiated from morph.rules) will be stored in an edge spanning the first and some following vertex in the chart. Phonological information about the stem will be stored in another edge, spanning the same vertices. Consequently, we may solve the problem by defining an operator with the effect of unifying the features of the lexical and the phonological edge, and assigning them to a new inactive edge of the same scope. This was our solution. Further details are given in /4/.

DICTIONARY SEARCH AND REWRITING

In Swedish morphological analysis, phonology plays a minor role. The idea of carrying out an independent phonological analysis to account for morphophonemic rewriting does not seem to be motivated. Cases that have to be handled concern e g

'umlaut' (cf fot/fötter), and secondary vowel (cf spegel/speglar).

Here, we would like to initiate the appropriate rewriting from signals in the input in connection with the dictionary search process. Such a strategy has been implemented in the following way.

Except for the ordinary entries, every dictionary contains four 'meta-entries', in which we specify how search in the dictionary should proceed. In the first entry we give the search key (e g CHAR). In the second entry we specify whether search for a longer key should be initiated when an entry has been found, or not. In the third entry we give the actions that should take place when a character has been found to match. This is the place where we may specify the signal for initiating processing in a certain grammar or grammar rule. As a matter of fact, this meta-entry specifies a grammar state itself, the rules of which are formulated in the UCP formalism. In our Swedish parser, we use this state for initiating processing of rewriting rules, part of the general Swedish grammar. Further details are given in /5/.

CURRENT APPLICATION

The current application of the UCP concerns the elaboration of a grammar and dictionaries for surface structure analysis of Swedish, including morphological and syntactic processing.

IMPLEMENTATION

The UCP is implemented in a subset of INTERLISP. It is being converted to LMI-CADRLISP. Except for the kernel system the implementation includes e g an editor for grammars and dictionaries. For details concerning the UCP formalism and the implementation, see /6/.

REFERENCES

/1/ Kay, M., Syntactic Processing and the Functional Sentence Perspective, in: Schank, R. and Nash-Webber, B.L. (eds.), Theoretical Issues in Natural Language Processing (TINLAP-1) (Cambridge, Mass., 1975).

/2/ Kay, M., Reversible Grammar, A Summary of the Formalism, Xerox PARC (1977).

/3/ Sågvall Hein, A., Finnish Morphological Analysis in the Reversible Grammar System, in: Proceedings from the 7th International Conference on Computational Linguistics (forthcoming).

/4/ Sågvall Hein, A., An Outline of a Computer Model of Finnish Word Recognition, in: Fenno-Ugrica Suecana, 3/1980 (Uppsala 1980)

/5/ Sågvall Hein, A., Uppsala Chart Parser Version 1 (UCP-1). - En översikt, in: Lien, E. (ed.), De nordiske datalingvistikkdagene 1981. Foredrag fra en konferanse på Universitetssenteret på Dragvoll 22.-23- oktober 1981 (Trondheim 1981) (in Swedish).

/6/ Carlsson, M., Uppsala Chart Processor 2, System Documentation, Uppsala University, Center for Computational Linguistics, Report No. UCDL-R-81-1 (1982).