Reports of Enlarged Session of the
Seminar of I. Vekua Institute
of Applied Mathematics
Volume 20, №3, 2005

SOFTWARE TOOLS FOR MORPHOLOGICAL AND SYNTACTIC ANALYSIS OF GEORGIAN TEXTS

Antidze J., Mishelashvili D.

I. Vekua Institute of Applied Mathematics

The "Software Tools for Morphological and Syntacßtic Analysis of Georgian Texts" is a software system designed for natural language, and particularly Georgian, texts processing. The system can be used to analyze syntactic and morphological structure of the natural language texts. Special formalisms were created for this purpose [1-2]. They allow us to write down syntactic and morphological rules defined by the particular natural language grammar. These formalisms represent a new, complex approach, to solve the problems connected with the natural language processing. A software system has been implemented according to these formalisms. Syntactic analysis of the sentences and morphological analysis of the word-forms can be done within this software system. Several special algorithms were designed for this system.

The system consists of two parts: syntactic analyzer and morphological analyzer. The purpose of the syntactic analyzer is to parse an input sentence, to build a parsing tree that describes relations between the individual words within the sentence, and to collect all important information about the input sentence that was figured out during the analysis process. It is necessary to provide a grammar file to the syntactic analyzer. There should be written syntactic rules of the particular natural language grammar in that file. Syntactic analyzer also needs information about the grammar categories of the word-forms of natural language. Information about the grammar categories of the word-forms will be used during the analysis process. However it may be quite difficult to include all of the word-forms from the natural language into a dictionary file. To avoid this problem, and to reduce size of the dictionary file, the morphological analyzer is used. The morphological analyzer uses a dictionary file of the static parts of words. Therefore this file will be considerably smaller, because many word-forms can be produced by single static part of the word. The morphological analyzer also needs its own grammar file. According to the special formalism, morphological rules of natural language must be written into that file. An input word can be divided between the morphemes when using these rules. And important information about the grammar categories can be deduced during the analysis.

An input sentence is passed to the syntactic analyzer. Syntactic analyzer will pass each word from the sentence to the morphological analyzer. Morphological analyzer will try to analyze the words according to the rules from the grammar file and the dictionary of words' static parts. After the successful analysis word-form will have information about all of the necessary grammar categories, and this information will be returned to the syntactic analyzer. Then the syntactic analyzer will try to parse the sentence according to the rules from the syntax file.

The basic methods and algorithms, that were used when developing the system, are: operations defined on the feature structures, trace back algorithm (for morphological analyzer), general syntactic parsing algorithm and feature constraints method.

Feature structures are widely used on all level of analysis. As an abstract data types they are used to hold various information about the dictionary entries. Each symbol defined within a morphological or syntactic rule has an associated feature structure, which is initially filled from the dictionary, or it is filled by the previous levels of analysis. Feature structures and operations defined on them are used to build up feature constraints. Feature structure is a set of features. Each feature is represented by a "name – value" pair. Feature may have a simple value (string) or a complex value (another feature structure). This is a recursive definition; therefore it is possible to define a feature structure which contains nested feature structures at any level. The square braces are used to write down feature structures according to our formalism. Feature name and value are separated by a colon. Examples:

[lex: dogs
cat: N
plural: +]
[f1: v1 f2: [f3: v3]]

The following operations are defined on the feature structures: equality check (=), unification ($\Leftarrow$), unification check (==) and assignment (:=). Each of them is a binary operation and after the completion returns a logical value, true or false. Equality check returns a true value if both feature structures have a similar feature names with similar corresponding values. Unification returns a true value if the feature structures does not have a similar feature names with different non-empty corresponding values. After the unification is done, all features that were in the second feature structure and were missed in the first one, are automatically transferred from the second feature structure. Unification check is equal to unification, but no automatic feature transfer is done. Assignment operation transfers the content of one feature structure into another, and always returns true value.

With general parsing algorithm it is possible to make a syntactic analysis of any sentence defined by context free grammar and simultaneously check the feature constraints that may be associated with any rule. Feature constraints are logical expressions build up with operations defined on the feature structures. The syntactic rules of a natural language are written into a grammar file as follows:

$S- > A_1\, A_2 ... A_N\, \{C\}$;

A_i are the symbols that form a RHS (right hand side) of the rule. C is feature constraints. There is also another form of a syntactic rule. It is possible to specify a custom order of the symbols within this form:

$S- > A_1\, A_2 ... A_N\, : R\{C\}$;

R is a symbol ordering regulators set. We can specify the symbol ordering rules in this set. I.e. $A < B$ means that the symbol A must be placed somewhere before the symbol B. $A - B$ means that the symbol A must be placed exactly in front of the symbol B. R is an empty set by default, in this case the program assumes that the symbols can be met in any order within this rule. In natural language it is a case when we have a free word ordering within the sentence. In example this form of a

rule is very convenient for Georgian language, because we avoid duplication of large number of rules to include all unique orderings of words. As we have noted, feature constraints can be attached to the rules defined into a grammar file. If the constraints are not satisfied during the analysis, then the current rule will be rejected and the search process will go on.

Feature constraints also can be attached to the morphological rules. However unlike the syntactic rules, constraints can be attached at any place within the morphological rule, not at the end only. This speeds up the morphological analysis, because the constraints will be checked as soon as they are met in the rule, and incorrect solutions will be rejected at the early stage. Morphological rules are written to the grammar file as follows:

$W- > M_1\{C_1\}M_2\{C_2\}...M_N\{C_N\ \}$;

M_i are morpheme classes. They must be defined before they are used in the rules. They are defined this way:

```
@m =
    {
    "aab" [feature1: value1],
    "bb"[],
    " " []
    }
```

Possible values of a morpheme class are written in double quotes. They are followed by the corresponding feature structure. "" is an empty morpheme. It means that this morpheme class may be omitted when dividing the word between the morphemes. C_i represents feature constraints. Trace back algorithm is used when dividing the word between the morphemes. It allows us to find all possible solutions, and check the feature constraints simultaneously when the analysis of the word is in progress.

The formalisms that were developed for the syntactic and morphological analyzers are highly comfortable for human. They have many constructions that make it easier to write grammar files. Morphological analyzer also has a built-in preprocessor, which has a capability to process parameterized macro insertions.

The software system is written in C++ programming language standard. It utilizes STL standard library. Program runs on UNIX and Windows operating systems. However it is possible to compile and run the program on any other platform where standard C++ compiler exists.

REFERENCES

1. Antidze J., Mishelashvili D. Instrumental tools for Georgian Texts computer analysis, Symposium on Georgian texts computer processing, Institute of Linguistics of Georgian Academy of Sciences, 2003, Tbilisi.

2. Antidze J., Mishelashvili D. Morphological Analyzer of Natural Languages, Conference on Natural Language Processing, The Georgian Language and Computer Technologies, Institute of Linguistics of Georgian Academy of Sciences, 2004, Tbilisi, Georgia.

Received 25. IV. 2005; accepted 20. X. 2005.