RULE-BASED INFLEXIONAL ANALYSIS

Zbigniew Jurkiewicz

University of Warsaw, Institute of Informatics
Palac kultury i nauki, 00-901 Warszawa, P.O.Box 1210, Poland

This paper presents a system for representation and use of inflexional knowledge for Polish language. By inflexional knowledge we mean information about rules of inflection/deflection for regular words together with a list of exceptions. Such knowledge can be successfully manipulated by a rule-based system. The research is a part of big undertaking, aimed at construction of a system able to converse in Polish with casual user.

The problem we are concerned with may be stated as follows. For each word in input sentence the system should find its basic form and dictionary information connected with it.

The simplest approach to this problem is to store all forms of words in the forms dictionary, which associates them with their basic forms. This method is acceptable for small sets of words. It places too big strain on system resources for bigger dictionaries. The way to minimize resource usage is to exploit regularities in the inflection.

Each language possesses some regularites in its inflexion. The extent of these regularities is different in different languages. Also the number of different inflectional forms may be different, e.g. an average polish verb can have about 100 forms. This forced us to think seriously about using regularities even in lexical components for small subsets of

of language. We view the inflectional analysis system as composed out of two parts:

- an exception dictionary with all forms taken as irregular,
- a mechanism exploiting regularities for getting necessary efficiency in search and saving resources.

We based our mechanism on the analysis of endings. The ending is defined as a part of word which is changed while reducing the word to its basic (dictionary) form. Polish language is characterized by many rules of deflection, which may be applicable to a given ending. A single word may be interpreted in as many ways as many endings we can distinguish in it, multiplied by a number of applicable rules for each ending. Therefore such candidate ending must be confirmed by checking result in the dictionary of basic forms after applying proposed deflection rule.

The described knowledge was written down in rule-based system "FORS". "FORS" is rather classical forward-driven rule system with some degree of extensibility. It is written in programming language LISP and is composed out of three parts:

- facts, represented as list structures stored in a fully indexed data base;
- rules of the form

  condition $\Rightarrow$ action action ...
- control mechanism for choosing and applying rules.

Each condition is a sequence of patterns of facts, which must be asserted in a database for rule to be applicable.

Patterns may contain typed variables. The type of a variable is identified by one-letter long prefix. Prefix must be a non-alphanumerical character. Variable type may be defined by providing matching functions for this type.

Inflexional knowledge is represented in "FORS" as follows. Each dictionary entry is represented as fact of the following form:

(ENTRY (BASIC-FORM) (CATEGORY) (OTHER PARAMETERS))

The word currently processed is saved as:

(RECEIVED(WORD))

The exceptions are represented as rules of the form

(RECEIVED (WORD-FORM)) (ENTRY (BASIC-FORM)...)
⇒ (ANSWER...)

The rules for deflection by endings replacement are stored as

(RECEIVED *VAR-(ENDING1)) (ENTRY *VAR-(ENDING2) ...) ⇒
(ANSWER...)

The prefix * is used for variables typed "suffixed". All variables in "FORS" get values by matching to fact elements. For suffixed variable without value, the value is assigned after cutting a given ending from item element (if possible, otherwise the matching fails). While matching suffixed variable which already has some value, Final value is obtained by concatenating given sufix to it.

There may exist many competing rules for recognized ending. Also, for a given word a couple of allowed endings may be indentified (e.g. one letter long, two letters long etc.). The control component in "FORS" allows to specify the sequencing between such completing rules. In a current version, the set of rules for regular endings is divided into groups according to the ending in

(RECEIVED...)

pattern. We associate a node with each such group. The nodes form a directed graph, called control graph. We associate a node with exception rules group too. One node is selected as a starting node. The arcs in this graph specify (partial)

order between nodes, thus defining sequencing between groups of rules. All nodes must be accessible from starting node (in other terms, control graph must be a directed acyclic connected graph).

The system works in cycles. At each cycle it reads the next word from input sentence and tries to find a rule applicable to this word. Rules are tried according to the order defined by a control graph, starting from the starting node. For each node, the rules associated with it are checked, until one is found with satisfied condition. This rule is then run and the next cycle begins. If no rule was applicable, system goes to one of successor nodes, guided by analysed word endings.

The advantages of representing inflectional knowledge in such a form are many. The system is modular, because each rule is independent from all others. Therefore rules may be added or deleted at will, allowing additional sources of knowledge to be tried.

The behaviour of the system is easily observable by non-programmer (in linguistic terms such as rules, endings etc.).

The set of rules may be adjusted to a given application, especially for small systems with specialised dictionaries.

The independent control component allows to experiment with different rule groupings in the search of minimization of resource usage. The grouping according to the concluded syntactic category may allow to exploit syntactic expectations, provided from parser. As for now, we succeeded in incorporating only most popular deflection rules (about 600 of them). We are going to incorporate some additional phonetic rules to take care of alterations. This could hopefully diminish the number of deflection rules.