Abstract

We present two experiments in progress concerning parsing Polish sentences, whose common ultimate goal is a powerful and efficient parser of Polish. The experiments are based on a metamorphosis grammar of a large subset of Polish; two parsers are involved in the experiments. One of them is written in Prolog. The other is implemented in ATN; it was obtained from the metamorphosis grammar using a straightforward algorithm. The parsers are being tested on the same set of about 100 sentences and phrases, comprising also the examples parsed by other parsers of Polish. We hope to demonstrate that the metamorphosis grammar formalism is a convenient tool for syntactic description of Polish and other Slavonic languages; the best way to convert the grammar into an efficient parser is still an open question.

Acknowledgement

This work would be much harder without the help of Stanisław Szpakowicz, who developed the metamorphosis grammar of Polish and made valuable suggestions concerning both the text and the improvements of the Prolog parsers.
1 A metamorphosis grammar of Polish

The linguistic theory behind the grammar was introduced by Saloni (1976) and further investigated in a series of papers which culminated in a book (Saloni, Świdziński 1981). The first approximation of the grammar was written down by Szpakowicz (1978); a revised version of the thesis, intended for linguists, has been published recently (Szpakowicz 1983). Some parts of the grammar were later rewritten in detail (Szpakowicz, Świdziński 1981, 1981a). The grammar is cast in terms of immediate-constituents analysis.

The formalism used is that metamorphosis grammars (Colmerauer 1975, 1978), sometimes called definite clause grammars (Pereira, Warren 1980). The core of the grammar, comprising about a half of the rules (i.e. about 200 rules), was debugged and tested as a program in Prolog (Szpakowicz 1978). The program can be used both to parse and generate sentences and phrases. The repertoire of syntactic constructions covered by the program is large enough to make it useful as a starting point in the design of linguistic components of dialogue systems.

2 A parser in Prolog

The original program was rewritten by S. Szpakowicz to serve as a general-purpose parser, which is now maintained and developed by J. S. Bień.

The program was modified in several ways. First of all, repeated computations were eliminated by factorization. Unfactored rules are more readable and linguistically intuitive, but usually inefficient in top-down parsers like those expressible in Prolog grammar rules. As an example of factorization, here are original rules for parsing compound sentences:

:ZDANIEZLOZ(*D) == :ZDANIESZER(*D) -/.  
:ZDANIEZLOZ(ZDANIEZLOZ(*D1,*D2,*D3,*D4)) == :ZDANIESZER(*D1)  

The same after factorization:

:ZDANIEZLOZ(*D) == :ZDANIESZER(*D1) :KZDANIAZLOZ(*D1,*D) -/.  
:KZDANIAZLOZ(*D1,ZDANIEZLOZ(*D1,*D2,*D3,*D4)) == :PRZEC(*D2)  

Secondly, the number of schemata analysed by the parser was increased by adding several new rules and updating several old ones. For example, the treatment of modifiers in the original program was restricted with respect to the full grammar (Szpakowicz 1978) — now some of the restrictions have been lifted. Some new rules were inspired by (Nalbach 1980).

Finally, dictionary lookup was reprogrammed by J. S. Bień. Since implementations of Prolog available to us do not support indexing of clauses, a dictionary
kept as one large Prolog relation is too costly. Instead, the dictionary was split into numerous small relations.

At the time of this writing, the augmented parser is operational but not yet fully debugged.

3 A parser in ATN

In this section, we shall describe how the metamorphosis grammar can serve as a starting point for a parser implemented in ATN.

As there is a natural correspondence between metamorphosis grammars and ATN, it is easy to construct an algorithm of translation. A recursive-descent method adapted to the case of non-LL-(1) grammars was used for this translation. The adaptation consisted in putting suitable conditions on arcs.

The parser is a collection of small networks. Each network corresponds to one non-terminal symbol of the grammar. It recognizes all the phrases or sentences which can be derived from the symbol. Its initial state is also named by this symbol. Separate ATNs communicate by means of PUSH arcs.

We shall now briefly describe how to construct one such ATN. The initial state of it corresponds to the left hand side of a production. If there is a terminal symbol on the right hand side of the production, a WRD arc is produced. If it is a category symbol, a CAT arc is created. In the case of a non-terminal symbol not equal to the lefthand side symbol a PUSH arc is created. When the lefthand side non-terminal symbol appears on the righthand side of a rule, a PUSH arc is not generated. Instead, a TO action allowing a direct transition to the initial state of the ATN, is created on the arc corresponding to the previous symbol of the righthand side of the production. It should be noted that the previous symbol always exists, if only there is no left recursion in the grammar. When symbols in the production are exhausted a POP arc is generated.

Parameters in metamorphosis grammar rules correspond to registers of the network.

The parser described in (Nalbach 1980) was written according to these rules and it did not include any other mechanisms. While observing traces produced by the running parser, the large amount of useless transitions was noticed. When the analysis failed, i.e. when there was no way to go farther from a given state (blocking state), the parser had to back up to the first state from which it could follow a different path. This caused useless blundering along the network and considerably increased the time of analysis.

According to these observations, conditions placed on arcs were extended to avoid improper transitions and to obtain a deterministic parser. The basic idea resembles that of Marcus (Marcus 1980). This notion can be described thus: wait and see if the transition you are going to do is really worth doing. The word “see” means here: look at current input word and, if it is needed, look forward at some of the following words to check if they satisfy all necessary conditions.

To implement this method some functions — look-ahead functions — had to be added to those of standard ATN function set.
The ATN parser of Polish was designed as a part of a system for natural language communication with relational data bases (Waligórski et al. 1980). In this application the parser should recognize mostly various kinds of questions and noun phrases functioning as questions. The language was first defined by a set of metamorphosis grammar rules. Most of the rules were based on those of (Szpakowicz 1978), but some new rules had to be added to parse various kind of questions and sequences of questions. In the first version of the parser some of the parameters were omitted for simplicity. However, most of them have been restored in the current version.

4 Implementations used in experiments

The parser in Prolog is being tested first of all on our principal mainframe implementation of Prolog — an interpreter written in Pascal (Kluźniak 1981). The latest release of this interpreter is available on IBM 370 and its Comecon-made analogues. As a very useful benchmark test, the parser will also be used on our minicomputer Prolog interpreter for SM-4 (compatible with PDP 11/40).

The parser in ATN uses an ATN compiler developed by Studziński. The compiler (for Polish-made MERA400 machines) generates assembler code. The whole ATN system is implemented in a relational data base environment, so that the ATN parser and the dictionary are relations in a data base.

Both parsers were originally implemented on a CDC Cyber 73, in Marseilles Prolog (Roussel 1975, Kluźniak 1984) and in an ATN interpreter written in Lisp (Studziński 1980).

5 Test data

To make our experiment comprehensive we decided to collect a wide spectrum of Polish sentences and phrases that have been quoted as parsed in papers discussing automatic analysis of Polish. Therefore, examples we have chosen (about a hundred in all) comprise not only test data of (Szpakowicz 1978) and (Nalbach 1980) but also a representative sample of sentences mentioned in (Bolc, Strzałkowski 1982, 1982a) and other papers pertaining to the DIALOG project.

A full list of test sentences and phrases, and parsing trees thereof, will be available from us on request. Here we give a few of them (also paraphrased in English) as an illustration of the diversity of test data:

Zarówno ojciec chciał, żebyśmy spali, jak też albo on albo ona spali.
(Both the father wanted us to sleep, and either he or she was sleeping.)

Czy jest pociąg do Kielc i o której on odjeżdża?
(Is there a train to Kielce and when does it depart?)

Wzrost napiecia mięśniówki dwunastnicy może być przyczyną wzrostu ciśnienia w przewodach trzustkowych.
(The increase of tonus of the tunica muscularis may cause higher pressure in the pancreatic ducts.)
6 Results expected

First of all, we intend to verify whether the parsers cover the test data; if not, we shall investigate the reasons. Secondly, we intend to gather quantitative data concerning the parsers’ performance. We hope to finish the experiments by September 1984.

7 Tentative conclusion

Our working hypothesis is that the metamorphosis grammar is general enough to handle most syntactic constructions which occur in existing systems with natural language interfaces, and flexible enough to be easily augmented if the need arise.

We hope to demonstrate that the metamorphosis grammar formalism is an excellent tool for precise and detailed syntactic description of (written) Polish and probably also other Slavonic languages. Its inherent limitations (cf Bień, Laus-Mączyńska, Szpakowicz 1980; Bień, Szpakowicz 1982) have only to do with highly marked constructions. In consequence, it can be used to advantage as a design specification.

There are two possible ways of converting such a design specification into an efficient parser. One is to completely rewrite the grammar; the target formalism may be e.g. pure Prolog or, as described in section 3, ATN. It should be noted that other strategies, e.g. bottom-up with look-ahead, are easily programmed in pure Prolog (Milne 1980). The other way is to transform the rules of the metamorphosis grammar; if interpreted, such a parser would be probably less efficient than a hand-coded program of the first approach, but an optimising compiler of Prolog could change the situation.

8 References


Bień J. S., Szpakowicz S.: Towards a Parsing Method for Free Word Order Languages. COLING 82 Abstracts, pp 34–41


Milne R.: *Parsing against Lexical Ambiguity.* COLING 80, pp 350–353


Saloni Z.: *Cechy składniowe polskiego czasownika.* Ossolineum 1976


Waligórski S. et al.: *Projekt i oprogramowanie modułu analizy języka naturalnego.* Institute of Informatics, Warsaw University 1980