An efficient implementation of a large grammar of Polish

Marcin Woliński

Instytut Podstaw Informatyki PAN
ul. Ordona 21, 01-237 Warszawa, Poland
wolinski@ipipan.waw.pl

Abstract

The paper presents a parser implementing Marek Świdziński's formal grammar of the Polish language. The grammar was written by a linguist, without a computational implementation in mind, unlike most formal grammars. The aim of the work reported here is to examine what the exact set of sentences accepted by the grammar is and what structures are assigned to them. For that reason, it was crucial to remain as close to the original grammar as possible. The resulting program, named Świgra, goes far beyond a toy parser due to the use of a morphological analyser and a broad range of linguistic phenomena covered by Świdziński's grammar. In this article preliminary results of the parser evaluation are also provided.

1. Introduction

So far, there have been only a few parsers of Polish and they usually do not cover a very large subset of the language. As far as we know, working parsers of Polish either build on Szpakowicz's grammar (Szpakowicz, 1983), which is rather small, or are based on non-formal grammars adapted for a given project. An interesting computational description of a subset of Polish was provided by (Przepiórkowski et al., 2001) but no efficient implementation was developed.

Świdziński's grammar, presented in his book (Świdziński, 1992, henceforth: GFJP), is probably the most extensive and most detailed formal grammar of Polish. Although it is expressed as a collection of formal rewrite rules, it was not intended for an implementation. And so, although the grammar was written over 10 years ago, it had no implementation before Świgra.

The domain of Świdziński's research is the so-called surface syntax of Polish, i.e., his grammar is free of any references to semantics. His aim was to describe all syntactic structures present in Polish, at least at the sentence level, as the lower phrase level is more sketchy in his grammar.

The goal of the work reported here is to implement this large formal grammar written by a non-computer-scientist and provide an evaluation of how close the description is to the Świdziński's intentions. The work has been the subject of the author's PhD thesis (Woliński, 2004).

The Świgra parser is based on earlier attempts to implement GFJP (Bień et al., 2000). Compared to these, it uses a new morphological analysis component and an improved parsing strategy. To achieve an efficient implementation, it was necessary to reformulate some of the grammar rules. Due to these changes, Świgra works with different structures than GFJP. The output of the parser, however, is presented in such a way that it pretends to be generated by the original Świdziński's grammar.

2. Morphological analysis

As is quite obvious for a language with rich inflection, the syntactic part of Świgra does not work with raw text but with wordforms resulting from a morphological analysis. The morphological analyser used in Świgra is *Morfeusz*, a program developed by the author of this paper, which uses linguistic data provided by prof. Z. Saloni[1]. The program uses a tagset developed for the corpus of Polish created at the Institute of Computer Science PAS (the IPI PAN Corpus). The tagset is based on the notion of a flexeme (proposed by Bień, 1991) — a set of wordforms which is uniform with respect to inflection (Przepiórkowski and Woliński, 2003; Woliński, 2003).

Morfeusz assigns to each token a lemma and a tag. More precisely, it tokenizes the input text and then assigns to each token all possible morphological interpretations, consisting of an identifier of the lexeme the form belongs to, and a tag describing the flexeme within a given lexeme and values of relevant grammatical categories.

More interestingly, the analysed text is not treated as a list of (lists of) possible interpretations, but as a directed acyclic graph (DAG, cf. Fig. 1). Nodes in the graph represent positions in the text (between tokens) and edges represent possible token interpretations. Edges are labelled with triples consisting of a token, a lemma and a tag. Edges of the DAG are represented as Prolog clauses.

In such a formulation of the parsing problem, a DAG is accepted by the grammar if and only if there exists a path in the graph labelled with a sequence of wordforms that can be derived from the grammar's starting symbol.

A similar idea is used by (Obrębski, 2002) but in his work wordforms are represented as labelled nodes and edges show a possible succession of nodes.

A graph is a convenient representation of ambiguities resulting in the morphological analysis. In Polish, a word segmentation itself can be ambiguous. For example the sequence of letters *ktoś* can be interpreted either as one token, a form of the lexeme SOMEBODY, or as a sequence of the token *kto* (lexeme WHO) and the token *ś*, a special form of the verb TO BE. This treatment is necessary to obtain a meaningful interpretation for the following Polish sentence:

[1]See `http://nlp.ipipan.waw.pl/~wolinski/morfeusz`. The program is available for download and can be used for non-commercial research purposes.

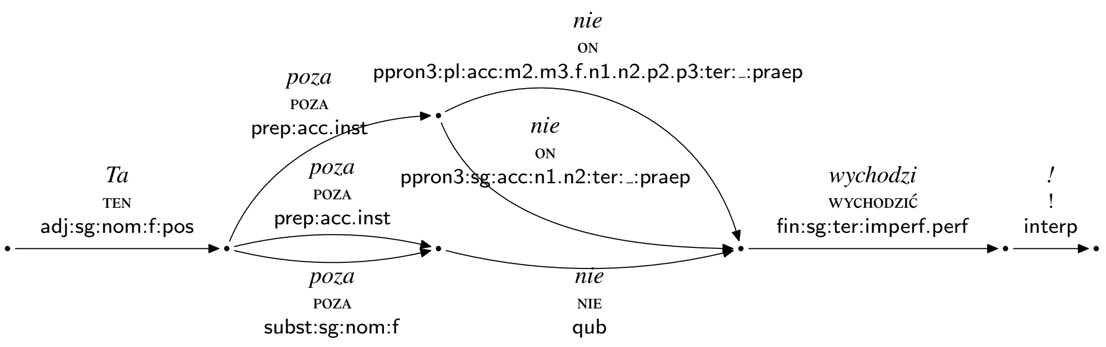

Figure 1: Morphological interpretations for the sentence *Ta poza nie wychodzi*.

(1) Ktoś ty?
Who-be.2PRS you?
'Who are you?'

A DAG is also a natural way of representing contextual dependencies between morphological interpretations (encoding information that some interpretation of a token is only possible if an adjacent token is interpreted in a particular way). A reasonable preliminary step for parsing would be to use a rule-based tagger that partially disambiguates the text by removing only truly impossible sequences of interpretations.

This has not been done in Świgra yet. However, a simple contextual mechanism has been introduced. The mechanism deals with Polish forms which can occur in texts only after a preposition. An example is *niego*, a form of the personal pronoun *on* (he). The corresponding form used in other contexts is *jego*. One of "postprepositional" forms of the pronoun *ona* (she) is *nie* which can be also interpreted as a negative particle (not). Unfortunately the problem of "postprepositionality" has not been considered by Świdziński. This leads to excessive structures assigned by GFJP to many sentences involving negation.

Ideally, the problem should be dealt with in the grammar but a quick partial solution is to contextually postprocess results of the morphological analysis. The idea is to change the input graph to put a "postprepositional" form with the preceding preposition on a path that is separated from other interpretations of the tokens.

An example is provided in Figure 1. It depicts a sentence that has two possible readings:

(2) Ta poza nie wychodzi!
This pose not works
'This pose doesn't work!'

(3) Ta poza nie wychodzi!
This beyond them leaves
'This goes beyond them!'

In the graph, *nie* interpreted as a postprepositional pronoun *she* is coupled with the interpretation of *poza* as a preposition 'beyond' (the upper path). In the lower path, *nie* is interpreted only as a negative participle but *poza* is allowed both as a preposition and a noun.

3. Syntactic analysis

The Świdziński's grammar is expressed in a formalism inspired by Colmerauer's metamorphosis grammar (Colmerauer, 1978), now more commonly known as Definite Clause Grammar (Pereira and Warren, 1980).

Although the formalism is close to Definite Clause Grammar available in Prolog implementations, the grammar cannot be implemented in a straightforward manner because of: missing rules of the lowest "preterminal" level, a void recursion (cycles of nonterminal elements which can be rewritten to each other), and an extensive use of conditions referring to values which have not been computed yet.

Moreover, rules describing an elementary sentence needed thorough clarification to get any reasonable results. This part of the grammar uses an extension of DCG that allows for permuting phrases constituting the elementary sentence, which is necessary due to the (relatively) free word order of Polish.

Świgra uses a bottom-up parsing strategy, which for Polish proved to be superior to the top-down strategy. The parser builds a shared parse forest (Billot and Lang, 1989), which is not only the result but also a means of avoiding unnecessary recomputation. In Świgra's strategy, a rule of the grammar may be called only as a consequence of adding a new edge to the forest. Since the number of edges in the parse forest is $O(n^2)$ (where n is the size of the input graph), this limits the computation time to polynominal, namely $O(n^{k+1})$, where k is the length of the longest right-hand side of a rule.

The key point for effectiveness is that rules are not interpreted at the runtime by the parser but they are compiled to Prolog clauses. This translation is somewhat similar to that used in the commonly quoted BUP parser (Matsumoto et al., 1983).

Thanks to the rule compilation, the parser proved to be more efficient for GFJP than a naïve chart parser. A chart parser has the computation time in the order of $O(n^3)$, so our algorithm is equally effective for grammars in Chomsky's normal form (which have at most 2 elements at the right hand side of a rule). In fact, we plan to extend our algorithm so it converts the rules to the Chomsky's normal form behind the scenes (as is commonly done in chart parsers).

Our algorithm differs from chart parsing since no in-

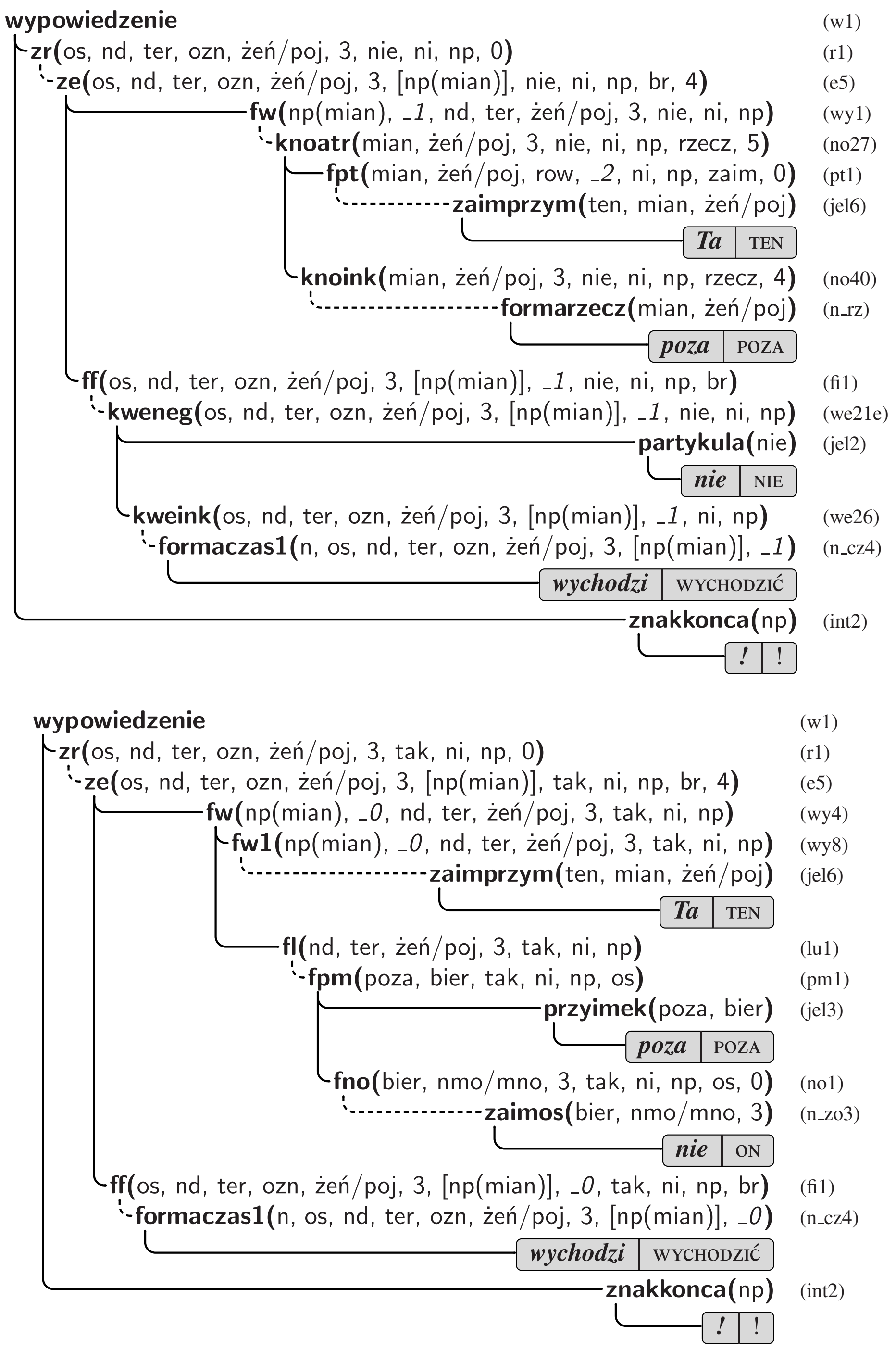
wypowiedzenie (w1)
zr(os, nd, ter, ozn, żeń/poj, 3, nie, ni, np, 0) (r1)
ze(os, nd, ter, ozn, żeń/poj, 3, [np(mian)], nie, ni, np, br, 4) (e5)
fw(np(mian), _1, nd, ter, żeń/poj, 3, nie, ni, np) (wy1)
knoatr(mian, żeń/poj, 3, nie, ni, np, rzecz, 5) (no27)
fpt(mian, żeń/poj, row, _2, ni, np, zaim, 0) (pt1)
zaimprzym(ten, mian, żeń/poj) (jel6)
Ta TEN
knoink(mian, żeń/poj, 3, nie, ni, np, rzecz, 4) (no40)
formarzecz(mian, żeń/poj) (n_rz)
poza POZA
ff(os, nd, ter, ozn, żeń/poj, 3, [np(mian)], _1, nie, ni, np, br) (fi1)
kweneg(os, nd, ter, ozn, żeń/poj, 3, [np(mian)], _1, nie, ni, np) (we21e)
partykula(nie) (jel2)
nie NIE
kweink(os, nd, ter, ozn, żeń/poj, 3, [np(mian)], _1, ni, np) (we26)
formaczas1(n, os, nd, ter, ozn, żeń/poj, 3, [np(mian)], _1) (n_cz4)
wychodzi WYCHODZIĆ
znakkonca(np) (int2)
! !
wypowiedzenie (w1)
zr(os, nd, ter, ozn, żeń/poj, 3, tak, ni, np, 0) (r1)
ze(os, nd, ter, ozn, żeń/poj, 3, [np(mian)], tak, ni, np, br, 4) (e5)
fw(np(mian), _0, nd, ter, żeń/poj, 3, tak, ni, np) (wy4)
fw1(np(mian), _0, nd, ter, żeń/poj, 3, tak, ni, np) (wy8)
zaimprzym(ten, mian, żeń/poj) (jel6)
Ta TEN
fl(nd, ter, żeń/poj, 3, tak, ni, np) (lu1)
fpm(poza, bier, tak, ni, np, os) (pm1)
przyimek(poza, bier) (jel3)
poza POZA
fno(bier, nmo/mno, 3, tak, ni, np, os, 0) (no1)
zaimos(bier, nmo/mno, 3) (n_zo3)
nie ON
ff(os, nd, ter, ozn, żeń/poj, 3, [np(mian)], _0, tak, ni, np, br) (fi1)
formaczas1(n, os, nd, ter, ozn, żeń/poj, 3, [np(mian)], _0) (n_cz4)
wychodzi WYCHODZIĆ
znakkonca(np) (int2)
! !

Figure 2: Two of parse trees for the sentence *Ta poza nie wychodzi!*

active edges are kept in the chart/forest. They are hidden in the recursion stack and reclaimed when no longer used, which provides for less memory requirements of the algorithm.

Results of the syntactic analysis are presented as sets of parse trees. For example Figure 2 depicts two of the trees generated by the program for the ambiguous sentence discussed in the previous section. These two trees correspond to the intended readings presented in (2) and (3). In the first *poza* is a noun (**formarzecz**) and *nie* is the negative particle (**partykula(**nie**)**). In the second sentence *poza* is a preposition (**przyimek**) and *nie* a personal pronoun (**zaimos**). In both cases the utterance (**wypowiedzenie**) consists of a coordinated sentence **zr** and final punctuation **znakkonca**. And the elementary sentence **ze** consists of a finite phrase **ff** and a required phrase **fw**.

The trees are shown in a short form where dotted lines represent collapsed non-branching paths in the tree. For example between the coordinated sentence (**zr**) and the elementary sentence (**ze**) there are three intermediate sentential units in the grammar.

Each nonterminal is presented together with its attributes, for example for the coordinated sentence **zr** the values are: personal os (form of verb), imperfective nd, present tense ter, indicative mood ozn, feminine żeń, singular poj, third person 3, not negated tak, without incorporation ni, and not a question np. An additional value for elementary sentence **ze** shows the types of required phrases, in this case one nominal phrase in nominative np(mian) (the subject).

4. Evaluation

The Świgra parser has been tested on 660 examples provided by Świdziński in his book. The set includes examples which are correct (should be accepted by GFJP according to Świdziński) and incorrect (should not be accepted). The main advantage of these examples is that information on correctness has been provided by Świdziński. The main disadvantage is that they are not extracted from real-life texts.

The following table presents results of parsing these sentences. The times and numbers of inferences are for building the shared parse forest representing all parse trees. The length of analysed sentences (including punctuation) varies between 3 and 27 (over a half of them in the range 7–16).

	correct		incorrect	
	acc.	nacc.	acc.	nacc.
sentences	515		145	
	469	46	34	111
	91%	9%	23%	77%
trees	10		14	
time (s)	0.26	0.36	0.20	0.15
inferences	267905	273601	201369	183400

Table 1: Results of parsing 660 sentences. The per sentence values for number of trees, time and number of inferences are medians. Time measured on a 1.8 GHz Pentium running Linux.

The longest time of an analysis is 302 seconds but 89% of the sentences are analysed within 1 second. The largest number of trees for a sentence is 2543013 (generated in 18 seconds).

It seems that almost all structures which Świdziński intended to describe are actually accepted by the parser — 91% of correct sentences are accepted. However somewhat large is the number of 23% incorrect accepted sentences. In many cases the problem is that the grammar accepts a sentence assigning it a completely different structure than intended by Świdziński. For example, one of the shortest incorrect examples accepted by the grammar is:

(4) Ona nie czytała książkę.
She not read book.ACC

In the correct form of the sentence the book is in genitive and not accusative case:

(5) Ona nie czytała książki.
She not read book.GEN

‘She wasn’t reading a book.’

In the interpretation provided by Świgra, the book is treated as in the following example, where a noun in accusative acts as a time complement. Such complements are described in GFJP as “free phrases”.

(6) Ona nie czytała godzinę.
She not read hour.ACC

‘She hasn’t been reading for an hour.’

Unfortunately, this is a limitation of the surface syntax — without resorting to semantics, it is impossible to reject sentence (4). The treatment of free phrases in GFJP allows them to attach in various places in the sentence structure. This often leads to accepting sentences of at least a doubtful grammaticality.

One of the reasons for the large number of parse trees produced by the program is morphological ambiguity. For example, in some sentences the gender of noun phrases can be limited to masculine but it is impossible to say which of the three masculine values (personal, animate or inanimate) should be used. Świgra uses a technique attributed to Colmerauer (Pulman, 1996) to represent such a set of genders as a single Prolog term. Moreover, the representation allows two terms to be unified only if the sets have a non-empty intersection, and the result of unifying them represents the intersection. In that way sets of values can be introduced to a unification grammar without modifying its rules. The experiments show that applying the same technique to the values of the grammatical case would lower the number of the resulting trees. However, not all cases of the spurious ambiguity could be eliminated in this way. A set of combinations of number, case and gender is often attached to a single token (e.g., in case of nouns and pronouns). The Colmerauer’s technique can be applied to any finite set, so it would be possible to use it for such combinations, but it would be necessary to modify the grammar rules to do that.

5. Perspectives

The parser should be seen as a work in progress. In the current stage the program is close to the original Świdziński’s grammar. We think that the grammar proved to be consistent enough to be a good starting point for improvements. However its limitations and deficiencies can clearly be seen. We now plan to start modifications of the grammar to limit superfluous interpretations and add missing features.

The most important extension of the grammar would be to incorporate rules for numeral phrases and coordination within phrases (e.g., nominal, adjectival). Both features are completely missing from GFJP. As for the linguistic data used by the parser, a more complete valency dictionary is needed, as well as a dictionary of multi-token lexical units.

For the overgeneration of parse trees, the problem of “free phrases” seems most important.

Świdziński’s grammar in its book form turned out to be much too complex and opaque to have a substantial impact on the linguistic audience. We hope that the existence of Świgra will encourage linguists to study the grammar which provides a deep insight into grammatical subtleties of Polish.

6. References

Bień, Janusz, Krzysztof Szafran, and Marcin Woliński, 2000. Experimental parsers of Polish. In *3. Europäische Konferenz "Formale Beschreibung slavischer Sprachen, Leipzig 1999"*, volume 75 of *Linguistische ArbeitsBerichte*. Institut für Linguistik, Universität Leipzig.

Bień, Janusz Stanisław, 1991. *Koncepcja słownikowej informacji morfologicznej i jej komputerowej weryfikacji*. Rozprawy Uniwersytetu Warszawskiego. Wydawnictwa Uniwersytetu Warszawskiego.

Billot, Sylvie and Bernard Lang, 1989. The structure of shared forests in ambiguous parsing. In *Meeting of the Association for Computational Linguistics*.

Colmerauer, Alain, 1978. Metamorphosis grammar. In Leonard Bolc (ed.), *Natural Language Communication with Computers*, Lecture Notes in Computer Science 63. Springer-Verlag, pages 133–189.

Matsumoto, Yuji, Hozumi Tanaka, H. Hirakawa, Hideo Miyoshi, and Hideki Yasukawa, 1983. BUP: a bottom-up parser embedded in PROLOG. *New Generation Computing*, 1:145–158.

Obrębski, Tomasz, 2002. *Automatyczna analiza składniowa języka polskiego z wykorzystaniem gramatyki zależnościowej*. Ph.D. thesis, Instytut Podstaw Informatyki PAN, Warszawa.

Pereira, Fernando and David H. D. Warren, 1980. Definite clause grammars for language analysis–a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13:231–278.

Przepiórkowski, Adam, Anna Kupść, Małgorzata Marciniak, and Agnieszka Mykowiecka, 2001. *Formalny opis języka polskiego. Teoria i implementacja*. Warszawa: Akademicka Oficyna Wydawnicza.

Przepiórkowski, Adam and Marcin Woliński, 2003. The unbearable lightness of tagging: A case study in morphosyntactic tagging of Polish. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03), EACL 2003*.

Pulman, Stephen G., 1996. Unification encodings of grammatical notations. *Computational Linguistics*, 22(3):295–327.

Szpakowicz, Stanisław, 1983. *Formalny opis składniowy zdań polskich*. Wydawnictwa Uniwersytetu Warszawskiego.

Świdziński, Marek, 1992. *Gramatyka formalna języka polskiego*. Rozprawy Uniwersytetu Warszawskiego. Warszawa: Wydawnictwa Uniwersytetu Warszawskiego.

Woliński, Marcin, 2003. System znaczników morfosyntaktycznych w korpusie IPI PAN. *Polonica*, XXII–XXIII:39–55.

Woliński, Marcin, 2004. *Komputerowa weryfikacja gramatyki Świdzińskiego*. Ph.D. thesis, Instytut Podstaw Informatyki PAN, Warszawa.