Świgra: An Implementation of a Large Grammar of Polish

Marcin Woliński

INSTITUTE OF COMPUTER SCIENCE
POLISH ACADEMY OF SCIENCES
ul. J. K. Ordona 21, 01-237 Warszawa

IPI PAN — April 19, 2006

Świdziński's grammar

Marek Świdziński

Gramatyka formalna języka polskiego, Wydawnictwa Uniwersytetu Warszawskiego, 1992.

- a grammar written by a linguist
- not intended for an implementation
- probably the most extensive and most detailed formal grammar of Polish
- so-called surface syntax (no semantics)
- expressed as a metamorphosis grammar
- 460 rules

The parser *Świgra*

Świgra is a parser written in Prolog implementing Świdziński's formal grammar.

The aim:

- to examine what the exact set of sentences accepted by the grammar is and what structures are assigned to them
- to remain close to the original grammar
- to verify Świdziński's description

The formalism

Metamorphosis grammars — a Prolog based formalism developed by Colmerauer:

Alain Colmerauer

Metamorphosis grammars
In: L. Bolc (ed.), Natural Language Communication with Computers. *Lecture Notes in Computer Science 63*. Springer-Verlag 1978, pp. 133–189.

Now more commonly known in the **Definite Clause Grammar (DCG)** variant proposed by Pereira and Warren (1980).

Morphological analysis

The parser does not work with raw tokens of the input text but with results of a morphological analysis.

Each token is equipped with morphological interpretations consisting of an identifier of the lexeme and values of relevant grammatical categories.

Morphological analyzer *Morfeusz*

- the program developed by Marcin Woliński
- linguistic data provided by prof. Zygmunt Saloni
- uses the IPI PAN tagset
- built using finite-state techniques

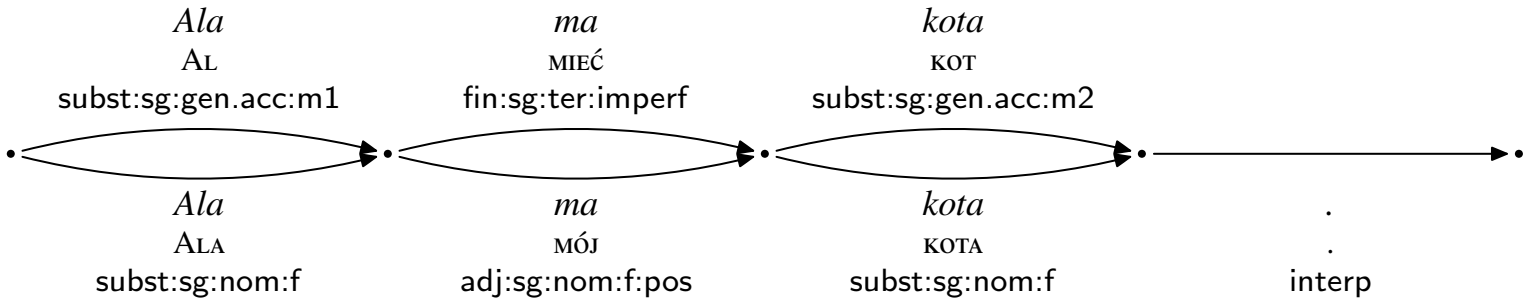
Morfeusz assigns to each token all possible morphological interpretations without checking context (it is not a tagger).

The tagset is based on the notion of a flexeme (proposed by Janusz Bień) — a set of wordforms which is uniform with respect to inflection.

Morphological analyzer *Morfeusz*

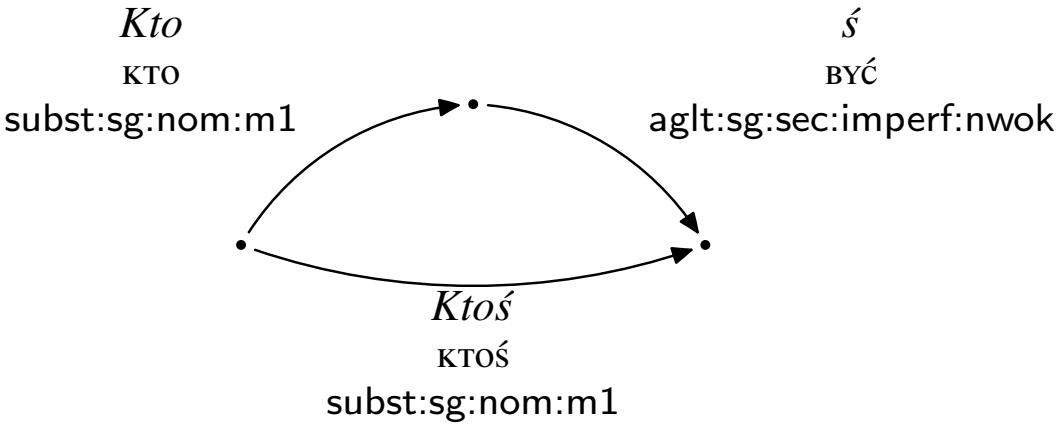
The analysed text is represented as a directed acyclic graph of interpretations:

- **nodes** — positions in the text (between tokens)
- **edges** labelled with possible token interpretations



In *Świgra* the DAG is stored in the form of Prolog clauses.

Word segmentation can be ambiguous



- Kto—ś ty?
  Who be.2PRS you

  ‘Who are you?’

- Ktoś przyszedł?
  Somebody came

  ‘Did anybody come?’

Morphological interpretations
for the sentence *'Ta poza nie wychodzi.'*

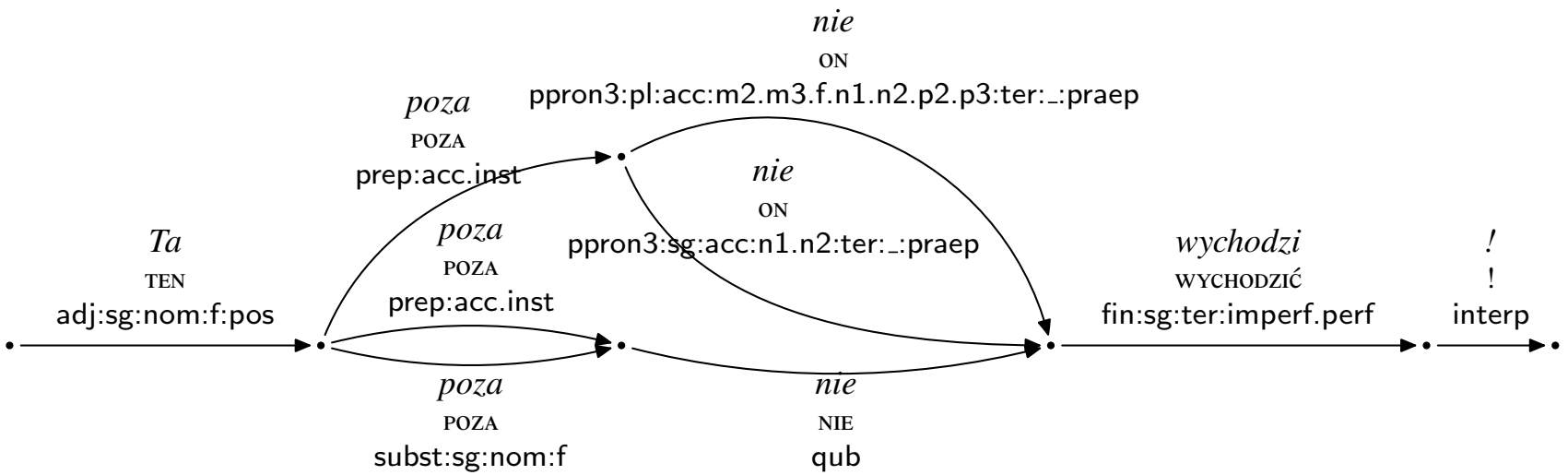
- Ta poza nie wychodzi!
  This pose not works

  'This pose doesn't work!'

- Ta poza nie wychodzi!
  This beyond them leaves

  'This goes beyond them!'

Morphological interpretations
for the sentence *'Ta poza nie wychodzi.'*



Syntactic analysis

Świdziński's grammar is close to the metamorphosis grammars formalism but:

- an extension is used that allows for permuting phrases,
- conditions often refer to values which have not been computed yet,
- void recursion is present (cycles of nonterminal elements which can be rewritten to each other),
- rules of the lowest "preterminal" level are missing,
- rules describing an elementary sentence needed thorough clarification.

The parsing algorithm

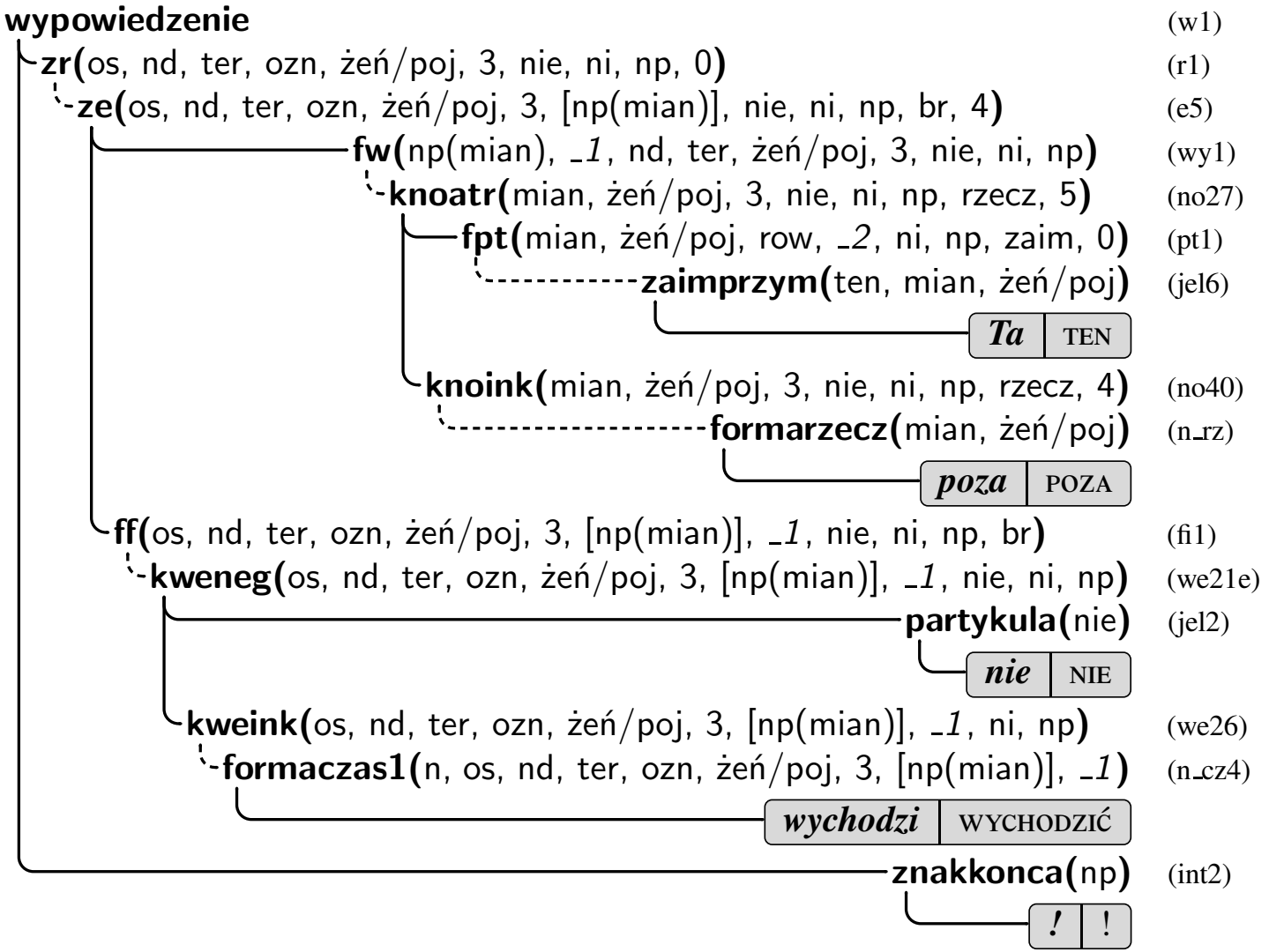
- Świgra uses a bottom-up parsing strategy, which for Polish proved to be superior to the top-down strategy.
- The parser builds a shared parse forest, which is not only the result but also a means of avoiding unnecessary recomputation.
- The rules of the grammar are not interpreted at the runtime but they are compiled to Prolog clauses.

The parsing algorithm

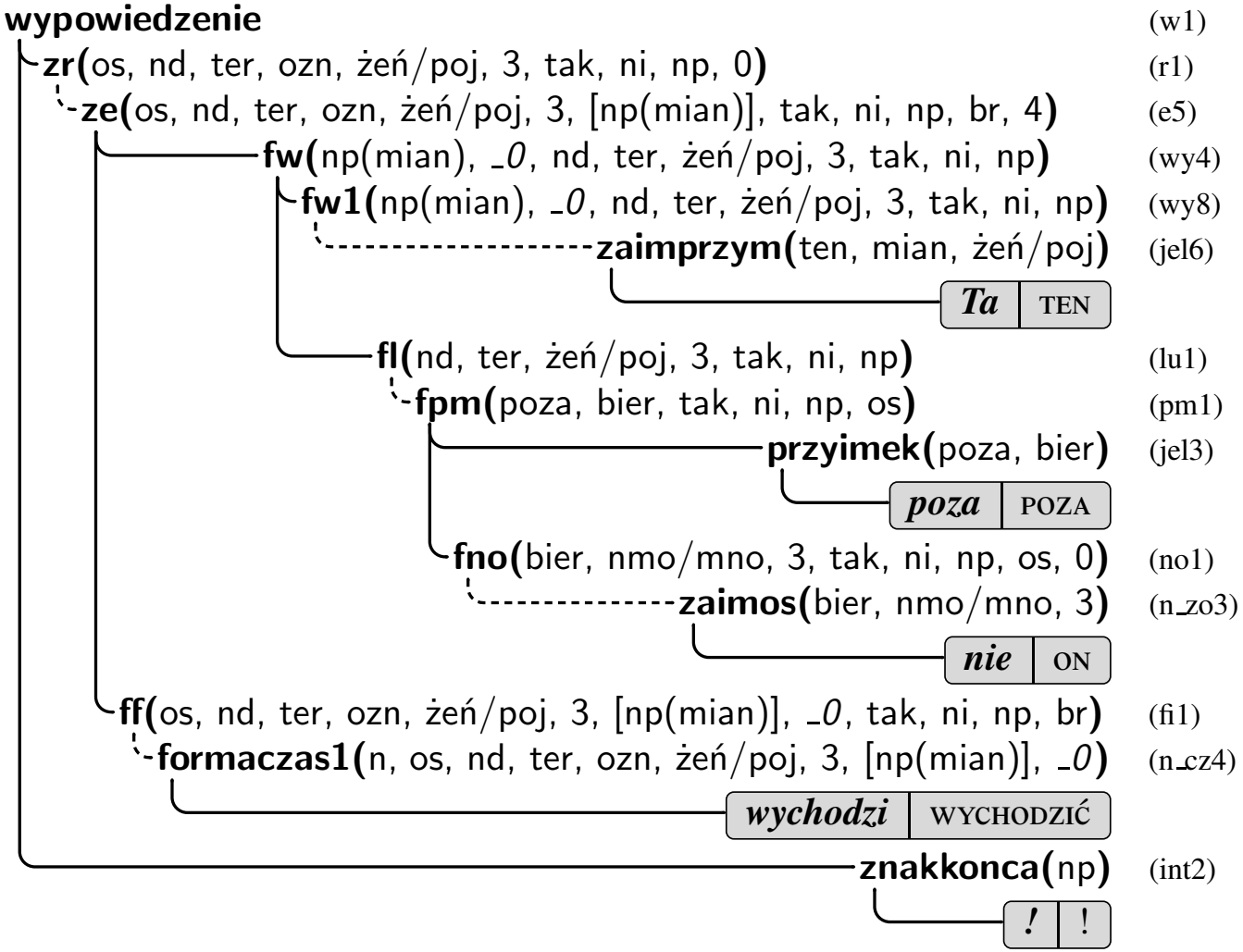
- Świgra uses a bottom-up parsing strategy, which for Polish proved to be superior to the top-down strategy.
- The parser builds a shared parse forest, which is not only the result but also a means of avoiding unnecessary recomputation.
- The rules of the grammar are not interpreted at the runtime but they are compiled to Prolog clauses.

Our algorithm differs from chart parsing since no inactive 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.

A parse tree


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)
! !

Another parse tree

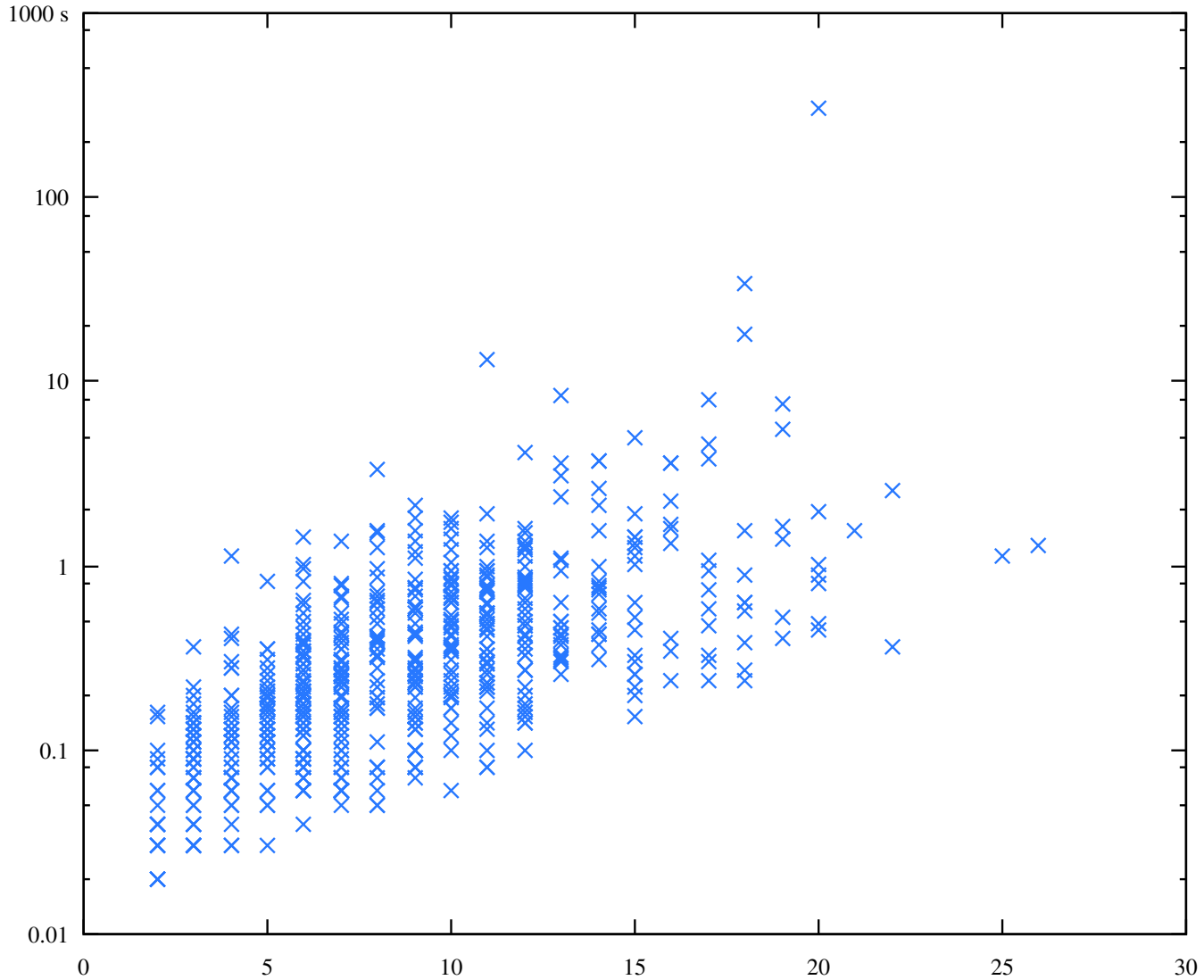


Preliminary results of evaluation

	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

Results of parsing 660 example sentences provided by Świdziński. Time measured on a 1.8 GHz Pentium running Linux.
The length of analysed sentences (including punctuation) varies between 3 and 27 (over a half of them in the range 7–16).

Parse time depending on sentence length



The extreme sentences

- *Mieliśmy czas, aż ani ja nie zostałem, ani on nie poszedł, ani ona nie przyszła.*
  1,029,654 trees, 18 seconds, 5861 edges in the forest
- *Gdybyśmy mieli czas, to ja nie zostanę, on nie pójdzie, ona nie przyjdzie ani dziecko nie uśnie.*
  302 seconds, 18,050 edges, 12,282 trees

Incorrect sentences accepted

One of the shortest "incorrect" sentences accepted by the grammar:

- Ona nie czytała książkę.
  She not read book.ACC

The correct form:

- Ona nie czytała książki.
  She not read book.GEN
  'She wasn't reading a book.'

Summary

- The grammar proved to be consistent enough to be a good starting point for improvements.
- Its limitations and deficiencies can clearly be seen.
- Extensions needed:
  - numeral phrases,
  - coordination within phrases (e.g., nominal, adjectival).
- Overgeneration of parse trees ("free phrases"!)

- *Świgra* is quite efficient.
- A parser implemented in Prolog provides for easy experiments with the grammar.
- *Świgra* is still experimental
  but already not a toy parser.