# On genitive clusters, Kleene star, and an exploding parser

**Tomasz Bartosiak, Marcin Woliński**

Institute of Computer Science
Polish Academy of Sciences
Jana Kazimierza 5, 01-248 Warszawa, Poland
tomasz.bartosiak@gmail.com
wolinski@ipipan.waw.pl

**Abstract**

The work reported here was triggered by an observation that the *Świgra* parser slows down on clusters of nouns in genitive that are present in some Polish sentences. The problem turned out to be connected with an extension to the DCG formalism used by the parser, namely with the ability to specify arbitrary sequences of nonterminals as part of the right hand side of some rules. In this paper we analyze this problem and its solutions, both theoretical and practical.

## 1. Introduction

In this paper we report on some experiences gathered while developing *Świgra*, a parser of Polish (Woliński, 2004; Świdziński and Woliński, 2010). The parser uses a DCG style grammar (Pereira and Warren, 1980). However, the word order in Polish is very flexible, which does not harmonize too well with the fixed order of constituents in DCG rules. Moreover, the grammar in question is mostly valence-driven. For example the constituents of a finite sentence (formed around a finite form of a verb) are fully determined by the valence dictionary. There can be from 0 to 7 arguments and an arbitrary number of adjuncts (for practical reasons limited to 3). At this point it is probably obvious that the trees we work with are not binary. It is generally agreed (Świdziński, 1992; Przepiórkowski et al., 2002) that binary trees typically used in grammars of English are not valid for Polish: the subject can easily appear in the sentence between, e.g., the verb and its object, so the view that the verb forms and its complements form an unbreakable phrase is difficult to defend.

We would be most happy to state in the grammar that a sentence consists of a verbal phrase and any permutation of dependents permitted by the valence dictionary for that verb. Unfortunately, this is impossible in DCG. For that reason we have developed an extension to the formalism that allows Kleene star to be used at the right hand side of grammar rules. A rule can take the form (simplified):

```
zdanie -->
    [fw(Type), fl]*,
    ff,
    [fw(Type), fl]*.
```

According to Świdziński's grammar (Świdziński, 1992) a sentence (`zdanie`) rewrites into a finite verbal phrase (`ff`) accompanied by sequences of arguments ("required phrases", `fw`) and complements ("free phrases", `fl`). The operator $\star$ means that any sequence of the enumerated elements can appear at that spot, in particular an empty sequence. The formalism provides a mechanism (not shown) to gather `Type`s of arguments and confront them with a valency schema taken from the valency dictionary. This matching happens immediately after a candidate argument is recognized, so in result the algorithm works as if all possible valence frames were hardcoded in the rules. Unfortunately, there are other constructs in which this extension proved useful where the number of constituents is not limited (an example of such constructions will be given in the next section). In the reminder of the paper we will examine problems in parsing caused by such unlimited sequences.

Work financed by the Polish Ministry of Science and Higher Education, a program in support of scientific units involved in the development of a European research infrastructure for the humanities and social sciences in the scope of the consortia CLARIN ERIC and ESS-ERIC, 2015-2016.

## 2. Genitive clusters in the grammar of Polish

Typical modifiers for a noun in Polish include adjectival phrases, nominal phrases in genitive (typically with a "possessive" meaning), and prepositional-nominal phrases. In this section we will take a closer look at structures that can be assigned to clusters of nouns in genitive. Such clusters can be quite long – the National Corpus of Polish (Przepiórkowski et al., 2011) contains sentences with as many as 11 nouns in the genitive in a row.

A typical structure for a genitive cluster is illustrated by the following example and Fig. 1:

(1) *Co* *jest* *konieczne* *do* *stwierdzenia*
what is necessary for declaring.GEN
*nieważności* *uchwały* *rady*
invalidness.GEN resolution.GEN council.GEN
*powiatu?*
county.GEN

'What is necessary to annul a resolution of the county council?'

Examples like (1) are most typical – the head (marked with a gray line in Fig. 1) is the left constituent at each level, while the right constituent acts as a genitive modifier. However, it is not the only possible structure. In the following example two modifiers are positioned to the left of the head:

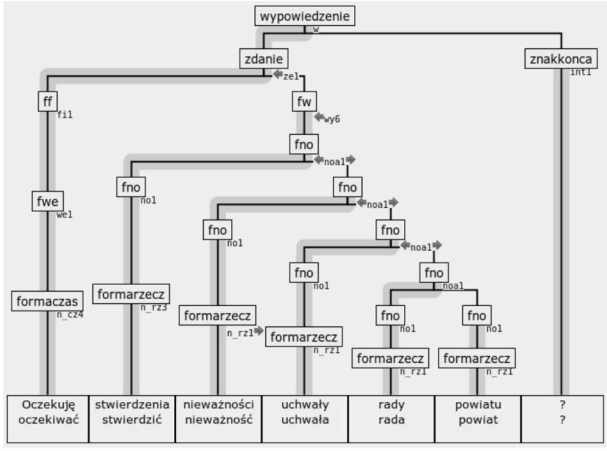


Figure 1: Typical right branching genitive cluster (cf. ex. (1))

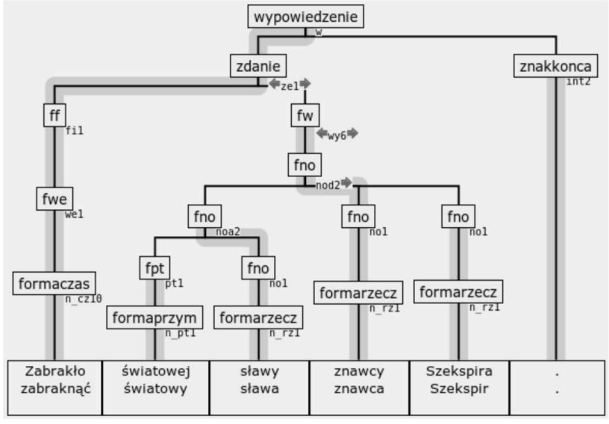


Figure 2: Genitive modifiers at both sides of the head (cf. ex. (3))

(2) *Szkoda jego świętej pamięci matki.*
pity he.GEN holy.GEN memory.GEN mother.GEN

'I pity his late mother.'

It is also possible for modifiers to appear both on the left and on the right (Fig. 2):

(3) *Zabrakło światowej sławy znawcy Szekspira.*
lacked world.ADJ.GEN fame.GEN expert.GEN Shakespeare.GEN

'The world-renowned expert on Shakespeare did not come.'

*Świgra* is being used to build the *Składnica* treebank of Polish (Woliński et al., 2011). The process involves automatic parsing and manual disambiguation of the resulting trees (or parse forests). For that reason we preferred the grammar to overgenerate – it is better for the annotators to choose from among too many trees than to omit the right structure from the parse forest. This way *Składnica* becomes a proof of coverage of *Świgra*.

For this reason the grammar includes a rule of the following general form:

```
np(Case) -->
    np(gen)*,
    np(Case),
    np(gen)*.
```

Any sequence of genitive modifiers is allowed on the left and on the right of the head of the rule, with additional condition (not shown) that enforces at least one of the two sequences to be nonempty to avoid void recursion. Moreover, the actual rule allows for other typical dependents of nouns within the sequences.

Unfortunately, we have observed that parsing sentences containing long sequences of genitives takes much more time than other constructs of similar length. Our hypothesis at this point was that the growth is exponential with the length of the genitive cluster.

## 3. Computational cost of sequences

While adding a Kleene star operator might seem a simple extension of CFG, it causes some problems for standard parsing algorithms. We will study its influence in this section.

Let's consider the following grammar that is a simplified model of what we apply to genitive clusters in *Świgra*:

```
A -> a
A -> A*
```

If we take into account strong generative capacity of the grammar, that is if we are interested in the shape of resulting parse trees, the above is equivalent to an infinite CFG:

```
A -> a
A -> A A
A -> A A A
...
```

In this section we will study performance on this grammar of three algorithms: a passive chart parser, an active chart parser, and finally an active chart parser with an extension for sequences.

### 3.1. Passive chart parser

*Świgra* presently uses a bottom-up passive chart parsing algorithm. Passive edges in a chart represent only completed phrases of various categories. This is enough to make the pessimistic parsing time polynomial (namely $O(n^{k+1})$, where $k$ is maximum length of the right hand side of a rule), while keeping memory consumption lower than for an active chart parser.

Unfortunately, for an "infinite CFG" parsing time becomes exponential. The length of the right hand side of a rule is limited only by the length of the input string (assuming no epsilon rules). So the computational cost becomes $O(n^{n+1})$.

A passive chart parser has to consider separately each way a parent in the tree can be split into a list of children. In the case of the perverse grammar in question this number is exponential. If we consider a sequence of $n$ letters `a`, simply each subsequence can be recognized as an occurence of

a nonterminal A. This means that at the topmost level the parser has to enumerate all splits of this sequence into 2, 3, ..., $n$ sub-sequences. Since the number of ways a sequence of $n$ elements can be split into $k$ sub-sequences is $\binom{n-1}{k-1}$, we get

$$\sum_{i=1}^{n-1}\binom{n-1}{i} = \left(\sum_{i=0}^{n-1}\binom{n-1}{i}\right) - 1 = 2^{n-1} - 1$$

different structures at this stage. This obviously results in exponential time for the whole algorithm (and exponential memory usage).

### 3.2. Active chart parser

An active chart parser stores not only completed phrases but also partially recognized right hand sides of rules. Such an algorithm is more memory-hungry, but its parsing time is $O(n^3)$.

An active chart parser internally converts the grammar into a form, where each rule has at most two elements at the right hand side. This is done with the so called dotted rules, which are in fact additional nonterminals in the grammar. Our simple grammar would produce the following rules:

```
A -> a

A -> [A -> AA.]
[A -> AA.] -> [A -> A.A] A
[A -> A.A] -> A

A -> [A -> AAA.]
[A -> AAA.] -> [A -> AA.A] A
[A -> AA.A] -> [A -> A.AA] A
[A -> A.AA] -> A

...
```

For input of length $n$ the process would result in $O(n^2)$ additional nonterminal symbols. This means that number of entries in each cell of the chart table is no longer limited by a constant, but can be as large as $O(n^2)$, which results in time complexity $O(n^7)$ for the whole algorithm.

It is worth noting that after parsing with converted rules the active chart parser has to reconstruct the trees that would result from the use of the original grammar. Active edges in the chart store parts of nodes of the resulting trees and not complete nodes. As we have seen in the previous section, the number of different realizations of a single node can get exponential. But the number of partial nodes stored in the chart remains polynomial.

### 3.3. Active chart parser tailored for sequences

Both algorithms considered so far worked on a grammar resulting from rewriting of the Kleene star into a (potentially) infinite series of grammar rules. The key observation to process sequences effectively is that we can treat the sequence as a special nonterminal and extend the parsing algorithm to deal with it directly.

First, it's worth noticing that some dotted nonterminal symbols in the CFG-equivalent are similar. For example [A -> A.AA] and [A -> AA.AA] are nonterminals stating that after getting 2 more As we will get A. This means that for sequences, it doesn't matter how many symbols have been detected so far. In fact, we can actually transform rules for chart parser into:

```
A -> a
A -> [A -> A^n .] for n > 1
[A -> A^n .] -> [A -> A^(n-1) .] A for n > 1
[A -> A.] -> A
```

Secondly, it doesn't really matter how many As have already been added to the sequence. [A -> $A^n$ .] and [A -> $A^m$ .] for different $n$ and $m$ are treated in exactly same way. That leads to set of four rules:

```
A -> a
A -> [A -> A.]
[A -> A.] -> [A -> A.] A
[A -> A.] -> A
```

This way the number of additional nonterminal symbols can be reduced to just one ([A -> A.]) – representing A⋆, a sequence of any length (this idea is present already in (Earley, 1970)). Active chart-parser based on such approach to Kleene star operator will work with time complexity of $O(n^3)$ while keeping memory consumption polynomial.

## 4. Experimental results

We have implemented the algorithm of Section 3.3. in *Świgra* and conducted experiments on genitive clusters of different length.

First we have used simplified grammar with only those rules, which are actually used for parsing genitive clusters and compared times of analysis and number of Prolog inferences for a parser with and without customized mechanism for Kleene star. Results of this experiment are presented (in logarithmic scale) in Fig. 3 and Fig. 4.

While results for the simplified grammar are promising, the same experiment conducted on the whole grammar shows that memorization of all sequences (like in an active chart parser) generates some problems. We clearly see that the number of inferences decreases greatly (Fig. 6) but the parsing time remains high (Fig. 5).

This is caused by a high constant hidden behind asymptotic value of $O(n^3)$. While the number of entries in any

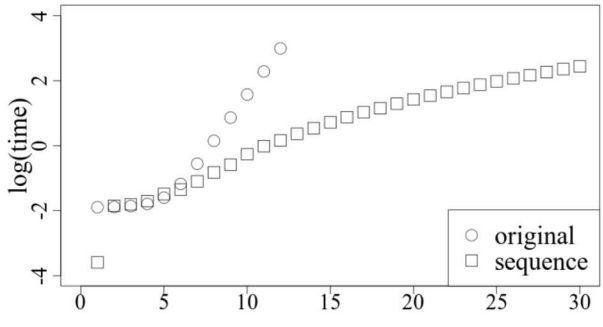


Figure 3: Parsing times for a grammar limited to clusters of genitives

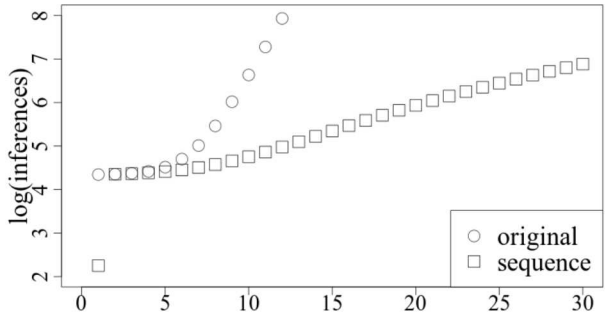


Figure 4: Number of Prolog inferences for a grammar limited to clusters of genitives

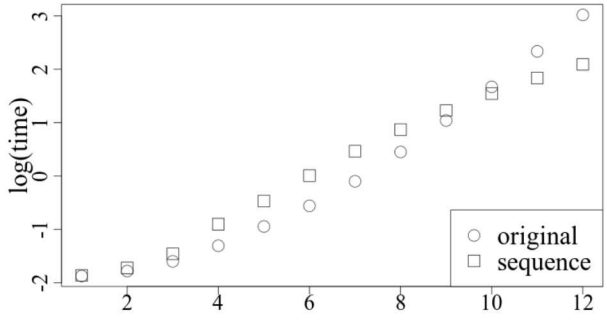


Figure 5: Parsing times for the full grammar

cell of the chart table is limited by a constant, the constant is so high, that the increase in time needed to search among entries outweighs the gain from storing partial results.

To further evaluate the solution we decided to run the parser with customized mechanism for Kleene star on the full corpus of *Składnica* treebank. We managed to reduce number of inferences below 45% of original number, but increased total parsing time by above 52% of the original.

Unfortunately, this means that the solution, which is perfect from the theoretical complexity point of view, does not offer any gain in the practice of parsing.

## 5. Limited rules for genitive clusters

Another question worth asking is whether unbound sequences are really needed to describe genitive clusters in Polish. Using a treebank search tool (Woliński and Zaborowski, 2012) we have analyzed all examples of genitive clusters longer than 2 elements present in the *Składnica* treebank (Woliński et al., 2011). We have also analyzed some longer sequences of genitives present in the full National Corpus of Polish.

The structure of modifiers positioned to the right of the head can be more complicated than a simple chain. However, possible branching seems to follow the schema illustrated in the following examples ((4) illustrated in Fig. 7):

(4) *Przybył* *sekretarz* *Sekcji* *Nauk*
arrived secretary section.GEN teachings.GEN
*Teologicznych* *Komisji* *Nauki*
theological commision.GEN teaching.GEN

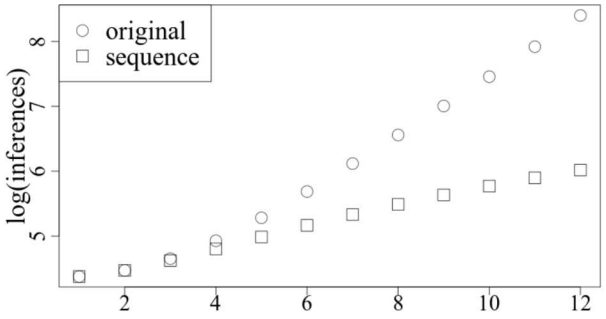


Figure 6: Number of Prolog inferences for the full grammar

*Wiary* *Konferencji* *Episkopatu*
faith.GEN conference.GEN episcopate.GEN
*Polski.*
Poland.GEN

'The secretary of the Section of Theological Sciences of the Doctrine Teaching Commission of the Polish Bishops' Conference has arrived.'

(5) *Dykcja* *większości* *jego* *wierszy* *przypomina*
diction most.GEN his.GEN poems.GEN resembles
*tę* *z* *przekładów* *Adama* *Ważyka*
that of translations.GEN Adam.GEN Ważyk.GEN
*wierszy* *Paula* *Eluarda.*
poems.GEN Paul.GEN Eluard.GEN

'Diction of most of his poems resembles that of Adam Ważyk's translations of Paul Eluard's poems.'

These structures generally contain chains of right modifiers, but at some levels there are two modifiers attached to the same head. In the case of example (4) the 'Section' is characterized both as a 'Section of Theological Sciences' and a 'Section of [some] Commision'. At the next level the 'Commision' is characterized as both pertaining to teaching faith and belonging to Bishops' Conference. The places where branching is possible are motivated by semantics. In this sentence the right pointing chains are names of some entities. This, however, need not be the case. In example 6 we have 'sb's translations of poems' with 'translations' and 'poems' being heads of respective genitival chains. We have not found any sentences where a phrase would have more than 3 children – as in the shown examples.

Nested modifiers to the left of the head are rare. They are marked and often sound pompous. Nevertheless, they cannot be ruled out:

(6) *Nie* *zaprzeczę* *słów* *jego* *doniosłości.*
not deny words.GEN he.GEN importance.GEN

'I cannot deny the importance of his words.'

The above analysis seems to support the belief that for practical purposes we can limit the number of genitival modifiers at any level to two (positioned arbitrarily with respect to the head).

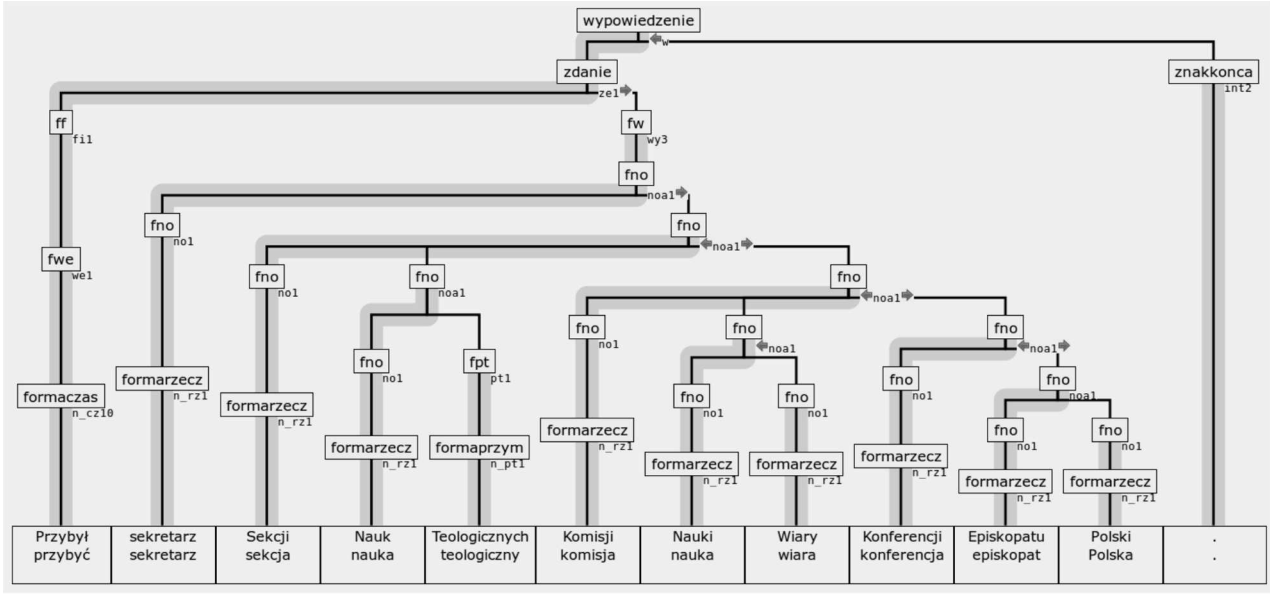


Figure 7: Heavily branching genitive cluster

Experiment with modified grammar shows better results than general mechanism for Kleene star. We managed to reduce number of inferences below $45\%$ of original number and parsing time to about $85\%$ of the original.

## 6. Conclusions

We have shown that parsing of unbound sequences can be improved to $O(n^3)$. Unfortunately, the larger "administrative" cost of the more complicated algorithm diminishes the gain in practical application. The improved algorithm outperforms the original only for sequences of genitives of about 10 elements or more, which are rare. The improvement is clearly visible when measured with the number of Prolog inferences, but when parsing time is measured the general results are worse than in the original implementation.

This again supports the decision taken in *Świgra* to use a passive chart parser as it in practice outperforms algorithms with better theoretical complexity.

The analysis of genitive modifiers in Polish shows that for practical reasons their number can be limited to 2, so unbound sequences are not needed in this case.

This, however, does not mean that Kleene star is not a useful extension. First of all, the number of adjectival and prepositional modifiers in nominal phrases seems not to be bounded in a similar way. Fortunately, these types of modifiers do not pose such problems in sequences because they cannot recursively modify themselves: an adjective cannot modify another adjective, it has to modify a noun (or a verb).

Moreover sequences are an elegant way to allow for an arbitrary number of arguments attached to a verb, as specified in the valence dictionary.

For that reason we plan to perform further experiments with the implementation to limit the amount of data being stored and in this way improve both speed and memory requirements of the algorithm.

## 7. References


Earley, Jay, 1970. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102.

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, Mirosław Bańko, Rafał L. Górski, and Barbara Lewandowska-Tomaszczyk (eds.), 2011. *Narodowy Korpus Języka Polskiego*. Warsaw: Wydawnictwo Naukowe PWN.

Przepiórkowski, Adam, Anna Kupść, Małgorzata Marciniak, and Agnieszka Mykowiecka, 2002. *Formalny opis języka polskiego: Teoria i implementacja*. Warszawa: Akademicka Oficyna Wydawnicza EXIT.

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

Woliński, Marcin, Katarzyna Głowińska, and Marek Świdziński, 2011. A preliminary version of Składnica—a treebank of Polish. In Zygmunt Vetulani (ed.), *Proceedings of the 5th Language & Technology Conference*. Poznań.

Woliński, Marcin and Andrzej Zaborowski, 2012. An ambiguity aware treebank search tool. In P. Sojka et al. (ed.), *TSD 2012*, volume 7499 of *LNCS*. Springer.

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

Świdziński, Marek and Marcin Woliński, 2010. Towards a bank of constituent parse trees for Polish. In Petr Sojka (ed.), *Text, Speech and Dialogue, 13th International Conference, TSD 2010, Brno, September 2010, Proceedings*, volume 6231 of *LNAI*. Heidelberg: Springer.