# Towards a Polish LTAG Grammar

Katarzyna Krasnowska

Institute of Computer Science, Polish Academy of Sciences,
k.krasnowska@phd.ipipan.waw.pl

**Abstract.** This paper reports on a Lexicalised Tree Adjoining Grammar for Polish, extracted automatically from the Polish constituency treebank. The grammar consists of 23 570 elementary trees anchored by 11 515 lexemes. Running the grammar on the sentences from the treebank using a modified version of TuLiPA parser showed that it achieves a high accordance (almost 99%) with the treebank annotation – in terms of syntactic categories assigned to phrases – on the trees which were successfully parsed. For many trees, however, obtaining a TAG parse was impossible due to time or memory shortcomings of the used tool.

**Keywords:** Tree Adjoining Grammar, treebanks, automatic grammar extraction

## 1 Introduction: LTAG grammars

This paper describes a Lexicalised Tree Adjoining Grammar for Polish, obtained automatically from the Polish constituency treebank *Składnica* [4]. Tree Adjoining Grammars (TAGs, see [2]) are a kind of tree rewriting formalism. A TAG grammar is formally defined as a quintuple: $\langle \Sigma, NT, I, A, S \rangle$, where $\Sigma$ is a finite set of terminals, $NT$ is a finite set of nonterminals ($\Sigma \cap NT = \emptyset$), $S$ is an initial symbol ($S \in NT$), $I$ and $A$ are finite sets of finite trees (*initial* and *auxiliary* trees). Initial and auxiliary trees have their internal nodes labelled with nonterminal symbols and leaves labelled with either terminals or nonterminals. A nonterminal leaf is a substitution site (usually marked "$\downarrow$"). Auxiliary trees have one special nonterminal leaf called foot node (marked "$*$" and labelled identically as the auxiliary tree's root). A Tree Adjoining Grammar is called *lexicalised* (LTAG) if each elementary tree has at least one terminal leaf. Such a leaf is by convention marked "$\diamond$" and called an anchor.

The trees are combined using two rewriting operations: substitution and adjunction (see Fig. 1). Derivation in a TAG grammar is a sequence of those operations, starting with an initial tree whose root is labelled with the initial symbol $S$. Tree substitution is fulfilled by attaching an initial tree to a nonterminal leaf. Substitution can be performed if the substitution node and the substituted tree's root have identical labels. Adjunction allows for insertion of auxiliary trees into the structure derived so far. For an adjunction to be possible, there must be an internal node (adjunction site) with a label identical as the adjoined tree's root (and, as follows from the definition of an auxiliary tree, its foot node). The adjunction site can then be replaced with the auxiliary tree.

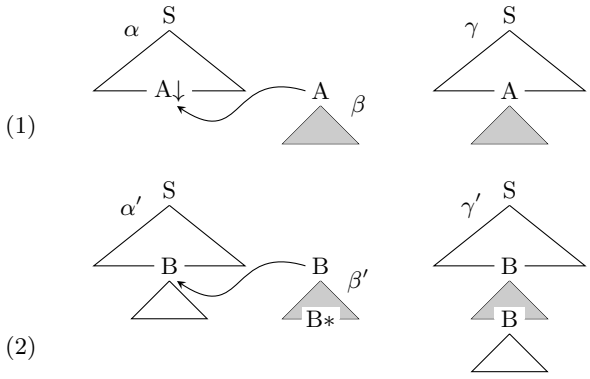



**Fig. 1.** Tree rewriting operations: (1) substitution of the initial tree $\beta$ into $\alpha$'s substitution node A↓, yielding $\gamma$; (2) adjunction of the auxiliary tree $\beta'$ into $\alpha'$'s internal node B, yielding $\gamma'$.

## 2 Extraction Procedure

The LTAG grammar extraction procedure is based on a technique proposed in [1], where such a grammar is obtained from the Penn Treebank. The extraction algorithm takes as its input a constituency tree and produces a set of elementary trees. It is a recursive procedure, starting in the root of the constituency tree. Extraction of an initial TAG tree $\alpha$ when the currently processed constituency tree node is $\eta$ is performed as follows:[1]

- make $\eta'$ — a copy of $\eta$ — $\alpha$'s root;
- for each non-head child of $\eta$, decide whether it is a complement or an adjunct;
- for each child of $\eta$, if it is:
  - a non-terminal head child, run the procedure recursively on it and attach its result as $\eta'$'s child;
  - a terminal head child, attach its copy as $\eta'$'s child and make it $\alpha$'s lexical anchor;
  - a complement, attach its copy as $\eta'$'s child and run the procedure recursively on it, producing a new initial tree;
  - an adjunct (to the node or its head child), run the procedure recursively on it and transform its result into an auxiliary tree as shown in Fig. 2;

The decision whether a child is a complement or an adjunct was taken according to rules such as the following:

- a node marked as mandatory phrase is a complement;

[1] The described extraction procedure requires knowing which of the current node's children is a head child [1] used a head percolation table to retrieve this information from Penn Treebank trees. In the case of *Składnica* it was not necessary since its trees have head children marked by design.

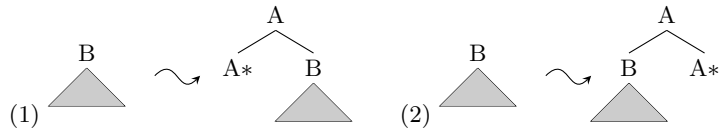



**Fig. 2.** Transformation of a tree extracted from A's adjunct B into an auxiliary tree (1) if B is a left adjunct (2) if B is a right adjunct.

- a node marked as a loose phrase is an adjunct;
- a node bearing a category label different from its parent's is a complement;
- other nodes are adjuncts.[2]

Marking of phrases as mandatory (fw, argument, according to valence dictionary) and loose (fl, modifier) is a feature of *Składnica* which is very useful for differentiating between complement and adjunct nodes. This marking appears, however, only at the level of the ff (main finite phrase) node's siblings (see an example *Składnica* tree in Fig. 3), therefore the other rules are also necessary.

*Składnica* is different from Penn Treebank in that its nodes contain not only a label representing the phrase's category, but also a set of morphosyntactic features.[3] This reflects the fact that Polish is a highly inflectional language. Some of the features appearing in *Składnica* were incorporated into the extracted elementary trees. Once an elementary tree was produced, its feature values were replaced with variables. The node features which should be in agreement (e.g. the gender features of a fwe – VP – node and its subject fno – NP – are assigned the same variable. Features which are required to have a specific value (e.g. the accusative case of an fno node representing an object) have this value explicitly specified in the tree. There were also some cases where *Składnica*'s way of handling morphosyntactic features had to be slighlty modified for the TAG grammar to work. For verbs which can appear in analytical form, but are only present in *Składnica* in non-analytical form (and vice versa), appropriate elementary trees were added. An example of TAG tree extracted from a constituency tree with feature values taken from the *Składnica* tree and after replacing those values with variables is shown in Fig. 3.

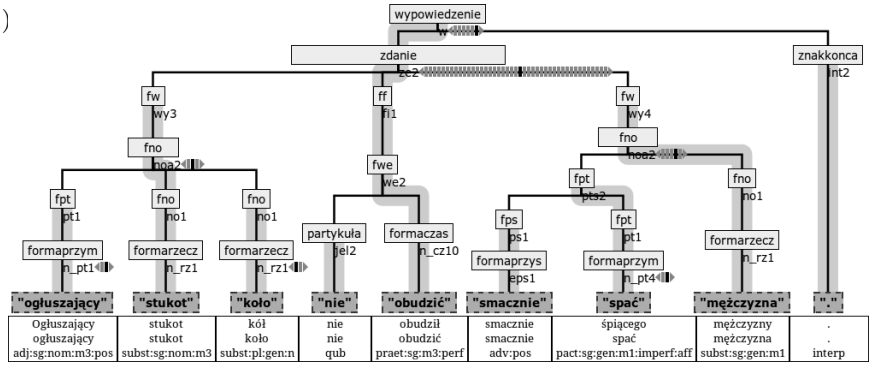
## 3 Parsing with TuLiPA

For the purpose of testing the grammar, the TuLiPA (The Tübingen Linguistic Parsing Architecture, see [3] and `https://sourcesup.cru.fr/tulipa/`) parser was chosen. TuLiPA allows for including features in elementary tree nodes and assigning variables to them in order to specify which features in the resulting tree should be equal. TuLiPA uses a 3-layer architecture with the lexicon divided into 3 files:

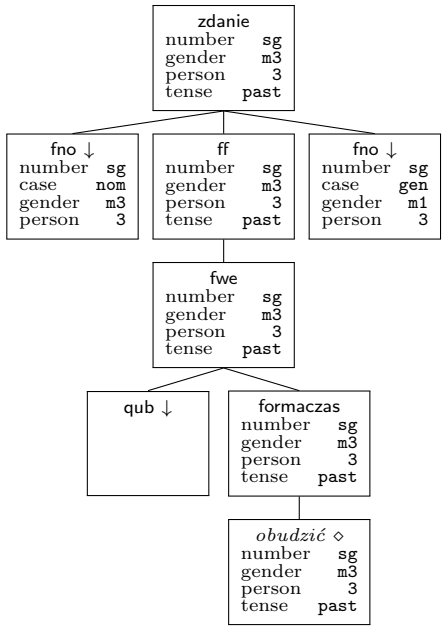
[2] To the head child if they are closer to it than any node's argument, to the node otherwise.

[3] Features included in the TAG grammar are number, case, gender, person and tense.

(1)




(2)




(3)

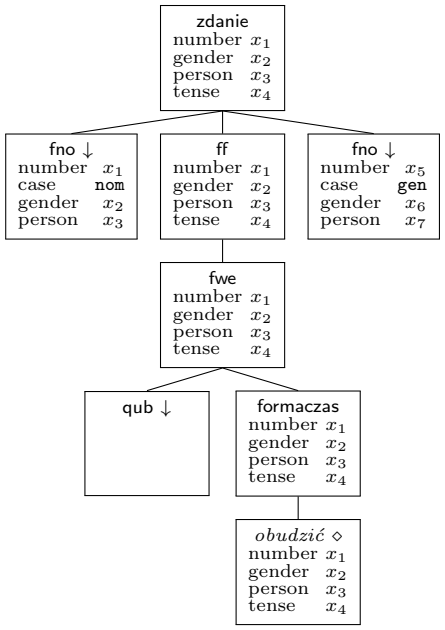



**Fig. 3.** (1) An example *Składnica* tree for the sentence *The deafening rattle of the train did not wake the soundly sleeping man up.* (node features hidden, highlighted branches lead to head children) and one of the TAG trees extracted from it: (2) with feature values from *Składnica*; (3) after replacing them with variables (its wypowiedzenie and znakkonca↓ nodes are not shown). This is an initial tree for the verb *obudzić* (*to wake*) in negated form, taking as its arguments a nominative subject and a genitive object.

– grammar containing tree families (elementary trees with "empty" anchors, i.e. no lexical elements specified);
– lexicon, a list of lexemes and corresponding tree family identifiers (each ⟨lexeme, tree family⟩ pair specified in this file defines a lexically anchored elementary tree);

– morphology, containing all possible morphological forms for each lexeme.

In the case of Polish TAG grammar, only the first two layers (grammar and lexicon) were produced. This was motivated by the fact that the lexicon contains all lexemes appearing in *Składnica*, and given the complexity of Polish inflection the morphology file would either be incomplete (if only the forms occurring in the treebank were included) or grow unreasonably large. Instead, TuLiPA was modified to use Morfeusz, a morphological analyser for Polish (see `http://sgjp.pl/morfeusz`). This modified version of the parser was called TuLiPA-pl. When given a sentence to parse, TuLiPA-pl produces a morphology file for it on the fly, comprising all morphological interpretations of the sentence's tokens given by Morfeusz. It is also possible to run TuLiPA-pl with a morphology file provided by the user, as in original TuLiPA.

## 4 Evaluation

The method described in Section 2 was applied to 7 229 sentences from *Składnica*.[4] The extracted grammar contains 2 802 elementary tree families (1 825 initial and 977 auxiliary). The lexicon contains 11 515 lexemes, anchoring a total of 23 570 elementary trees (one lexeme can serve as a lexical anchor to more than one tree, e.g. in case of verbs with more than one possible valence frame). The average number of trees anchored by a lexeme is 2.05; 7 953 lexemes (69%) anchor only one tree.

Preliminary tests with parsing using the produced Polish TAG grammar and TuLiPA-pl showed serious efficiency problems: even relatively simple sentences either took a very long time to be processed or caused to parser to run out of memory. To speed up TuLiPA-pl's performance for the purpose of grammar evaluation, morphology files (one per sentence, containing morphological interpretations of token chosen in the particular tree) were extracted from the treebank. In this way, the parser did not have to deal with ambiguous word forms.

The extracted grammar was used to run TuLiPA-pl on the same sentences from *Składnica* that were used for extraction. Even with unambiguous morphology files, parsing of many sentences was unsuccessful due to time or memory shortcomings. Out of 7 229 sentences, TuLiPA-pl managed to produce a parse forest for 2 806. Table 1 summarises the outcomes of those TuLiPA-pl runs.

For the sentences for which TuLiPA terminated its run (with or without a parse, a total of 2806 sentences), the trees produced by the TAG grammar were compared to the original *Składnica* trees. For each *Składnica* tree and for each parse in the corresponding parse forest (empty in the cases when there was no parse) generated by TuLiPA-pl, the number of phrases assigned the same category (dominating node's label) was calculated. Then, the best-matching tree (i.e. the one with the most matching phrase categories) was chosen from the forest. The percentage of all phrases from *Składnica* which were assigned the

[4] 998 trees were excluded due to technical problems they posed (mainly errors in head child marking).

**Table 1.** Outcomes of TuLiPA runs. Time limit was 5 minutes per sentence.

| outcome | parse | no parse | TuLiPA error | out of memory | timed out |
|---|---|---|---|---|---|
| sentences | 2678 | 128 | 640 | 3697 | 44 |
| percentage | 37% | 2% | 9% | 51% | 1% |

same category in the best-matching TAG parse was 92%. With sentences limited to the ones with a non-empty parse forest, this score was 98.8%.

## 5 Conclusions

A Tree Adjoining Grammar was extracted automatically from the Polish constituency treebank. Although the grammar perform quite well on the sentences it manages to parse, achieving an almost 99% phrase category assignment match with the treebank, there are performance issues which make the grammar (at least when TuLiPA parser is used) inefficient for parsing large amounts of text. Moreover, the TAG formalism does not seem very well suited for languages with loose word order such as Polish since it requires that the elementary trees have a fixed structure. It is nevertheless worthwhile to report on this experiment since it is, as far as we know, the first attempt at creating a wide-coverage TAG grammar for Polish.

*Acknowledgements* The work described in this paper is partially supported by the DG INFSO of the European Commission through the ICT Policy Support Programme, Grant agreement no.: 271022, as well as by the POIG.01.01.02-14-013/09 project co-financed by the European Union under the European Regional Development Fund.

## References

1. John Chen and K. Vijay-Shanker. Automated extraction of Tags from the Penn Treebank. In *Proceedings of IWPT 2000*, 2000.
2. Aravind Joshi and Yves Schabes. Tree-adjoining grammars. In *Handbook of Formal Lanaguages and Automata*. Springer-Verlag, Berlin, 1997.
3. Laura Kallmeyer, Timm Lichte, Wolfgang Maier, Yannick Parmentier, Johannes Dellert, and Kilian Evang. TuLiPA: Towards a multi-formalism parsing environment for grammar engineering. In *Coling 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks*, pages 1–8, Manchester, England, August 2008. Coling 2008 Organizing Committee.
4. Marcin Woliński, Katarzyna Głowińska, and Marek Świdziński. A preliminary version of Składnica—a treebank of Polish. In Zygmunt Vetulani, editor, *Proceedings of the 5th Language & Technology Conference*, pages 299–303, Poznań, 2011.