Shared forest representation of predicate-argument structures for shared syntactic forests

Tomasz Bartosiak

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

Abstract

This paper describes a method to create predicate-argument structure based on a shared syntactic forest. Since number of semantic interpretations can be exponential, we show a few mechanisms that can be used to share partial results and maintain both time and space complexity of whole procedure in reasonable bounds.

1. Introduction

In this article we describe a method to construct a predicate-argument structure based on a representation of syntactic parse trees of Polish utterances. It is the first step in creating full deep semantic description.

The most important source of information on predicates and their arguments for Polish is *Walenty* (Przepiórkowski et al., 2014; Hajnicz et al., 2016b; Przepiórkowski et al., 2017) – a syntactico-semantic valence dictionary. It is used by multiple syntactic parsers. One of them is *Świgra* (Woliński, 2004; Świdziński and Woliński, 2010; Woliński, 2015), which uses *Walenty* to determine possible syntactic dependants. As *Walenty* describes syntactic representation of semantic arguments, this allows to create a matching predicate-argument structures.

Our system uses shared forest representation produced by *Świgra* and creates all corresponding predicate-argument structures. Due to combinatorial explosion, resulting structures (or forest of those structures, to be more precise) should be created in form of shared forests.

The main goal of this article is to show how such shared predicate-argument structure might look like.

2. *Walenty* as an information source for predicate-argument structures

In traditional linguistics a *predicate-argument structure* describes states of affairs (situation, process, etc.) presented in an utterance. In this paper we will slightly widen that concept. First, we will create structures also for words describing the attitude of the speaker (we won't distinguish between personal belief of speaker (*Myślę, że* '*I think that*') and statements about beliefs of others (*Jan myśli, że* '*John thinks that*')). Second, we will construct structures for words that are not traditionally considered to represent a predicate, including e.g., type describing nouns.

We will call all semantic descriptions of words from an utterance that show semantic dependants – a predicate-argument structure corresponding to that word. All dependants in such descriptions should have their function (with respect to the primary concept) labelled.

Acknowledgment Works funded by the Polish Ministry of Science and Higher Education within CLARIN-PL Research Infrastructure.

As we have mentioned in section 1., predicate-argument structures will be based on *Walenty*. The dictionary has two connected layers – syntactic and semantic. In its syntactic layer all predicates have their dependants described with syntactic and morphosyntactic constraints. In the semantic layer (Hajnicz et al., 2016a) each of possible meanings of an entry has a list of semantic arguments it can take. Arguments have their function described as thematic roles. Furthermore, each argument has a typical conceptual content described in a form of selectional preferences. An important part of *Walenty* is a connection between those layers. It shows which argument and in what syntactic form can co-occur in an utterance.

An example of predicate-argument structure for senstence *Wino napełnia kieliszki* ('*Wine is filling glasses*'), based on thematic roles from *Walenty*, can be seen in Fig. 1 in the form of attribute-value matrix. SENSE attribute identifies the structure as a whole and can be seen as the type of this structure. Other attributes are thematic roles as assigned by *Walenty*.

$$
{}_{1}\left[\begin{array}{ll}
\textsc{Sense} & \textit{napełniać-1}\ \text{'fill'} \\
\textsc{Theme}^{\textsc{Source}} & {}_{2}\left[\begin{array}{ll}\textsc{Sense} & \textit{wino-1}\ \text{'wine'}\end{array}\right] \\
\textsc{Theme}^{\textsc{Goal}} & {}_{3}\left[\begin{array}{ll}\textsc{Sense} & \textit{kieliszek-1}\ \text{'glass'}\end{array}\right]
\end{array}\right]
$$

Figure 1: Simple predicate-argument structure.

In Fig. 1 we see that *wino-1* '*wine*' and *kieliszek-1* '*glass*' are arguments of a predicate *napełniać-1* '*fill*'. Thematic roles assigned to them in *Walenty* are $\textsc{Theme}^{\textsc{Source}}$ (the liquid being poured) and $\textsc{Theme}^{\textsc{Goal}}$ (the vessel being filled) correspondingly. There are multiple other arguments that were omitted in the sentence, e.g. person pouring the liquid (INITIATOR).

3. Anchoring predicate-argument structures in constituency trees created with *Świgra*

Construction of predicate-argument structures is based on syntactic parse forests. In our case, input would be

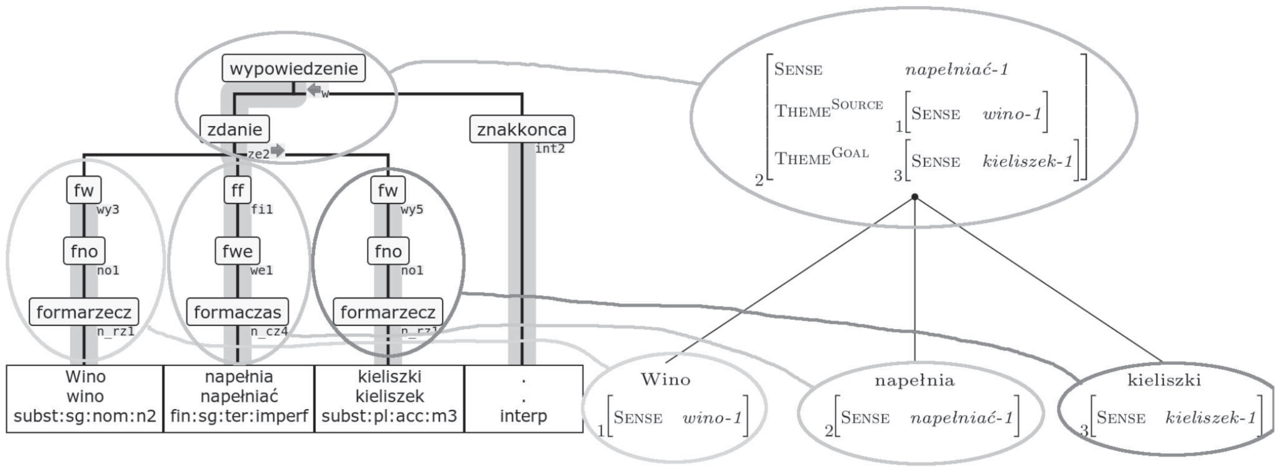


Figure 2: Correspondence between nodes of constituency tree and nodes of predicate-argument tree.

shared forests created with *Świgra* parser. The parser, stemming from metamorphosis grammar of Polish created by Świdziński (1992), generates constituency trees. *Walenty* is used to determine structure of dependants of predicates and to store information what syntactic structure do those dependants have (Woliński, 2015).

Since a single predicate-argument structure will be constructed based on a constituency tree, we will assign to each node a predicate-argument structure corresponding only to elements of sub-tree rooted in this node. E.g., we do not want to assign structure with complements to a node representing only the predicate. In Fig. 2 we connect simple structures representing concepts to nodes representing phrases consisting of only 1 word, while predicate-argument structure of whole sentence is connected to nodes spanning whole utterance. It results from composing constituent structures according to rules of dedicated semantic grammar.

Predicate-argument structures can be connected in a structure corresponding to syntactic trees. We will call such graph a predicate-argument tree. Later in this article we will ignore syntactic trees/forests and will focus only on corresponding predicate-argument structures.

4. Creating a predicate-argument forest

As we have mentioned in section 1., input for our algorithm is not a single tree, but a shared forest. Still, every grammar rule for creating predicate-argument structure takes into account single production from the shared forest. As result each structure corresponds not only to a syntactic forest node, but also to a rule that was used to create it.

Let us look at a predicate-argument forest created on the base of a single tree (a forest with a single tree) with all words additionally sense disambiguated (Fig. 3) corresponding to a sentence *„Skwaśniałe wino napełnia kieliszki."* (eng. *"Sour wine fills glasses."*).

Even with such strong constraints (single tree, word sense disambiguated) we produce multiple predicate-argument structures corresponding to the root node. What is more problematic, those structures, while corresponding to a single node of syntactic forest, cannot be treated equally (i.e., they cannot be used interchangeably for any further tree processing; more precise explanation in section 5.).

In reality situation is usually worse. Shared forests consist of multiple trees and words appearing in an utterance have multiple meanings that share syntactic structure. This highly increases amount of data that needs to be processed and stored compared to the syntactic forest. This calls for some sharing/packing algorithm adapted specifically for predicate-argument forests.

5. How to reduce amount of stored data

Let us analyze, how we can reduce the amount of stored data. First of all, we can notice that, as we are getting closer to the root, we are using the same structures that are simply more instantiated. We can use this fact to replace representation of a whole predicate-argument structure with its number and information about attribute values assignments performed.

The real complexity problem comes from the fact that a single node in a syntactic forest can correspond to multiple predicate-argument forest nodes. Let us consider sentence *Niepokój napełnia duszę i dręczy serce* ('Anxiety fills the soul and torments the heart'), sense disambiguated predicate-argument forest of which is presented in Fig. 4.

There are two nodes corresponding to a single node from constituency tree representing phrase *napełnia duszę*. Those structures are created due to different semantic interpretations of the same syntax. Any further arguments added to those structures depends on how the structure was constructed. This causes a real problem, as from the point of view of the syntactic grammar this seems to be context-sensitive information.

Such situation can be avoided with knowledge about used syntactic schemata. Once we know exactly which syn-

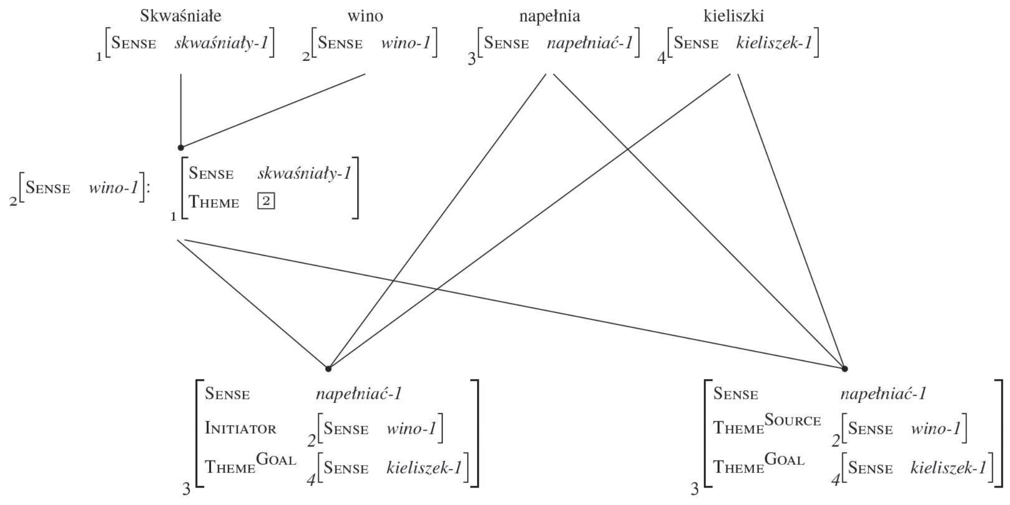


Figure 3: Predicate-argument forest created for a word sense disambiguated syntactic tree.

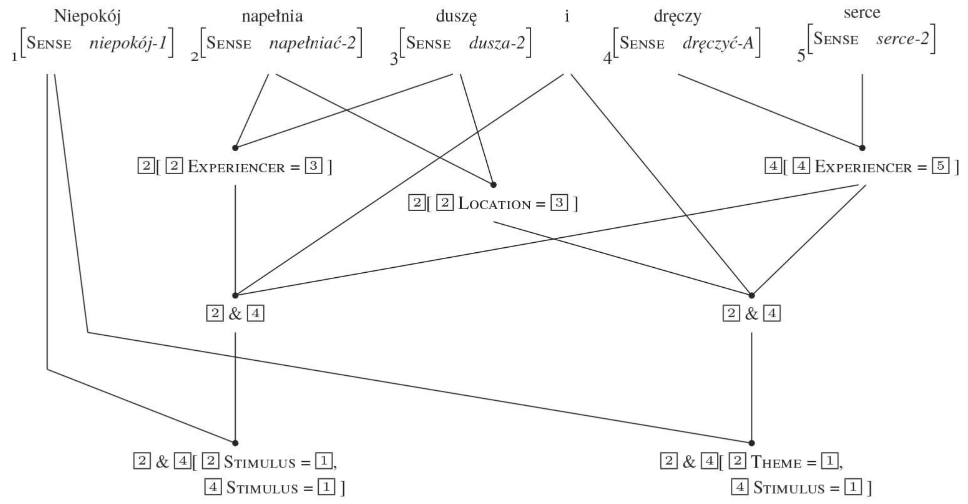


Figure 4: Predicate-argument forest, difference notation.

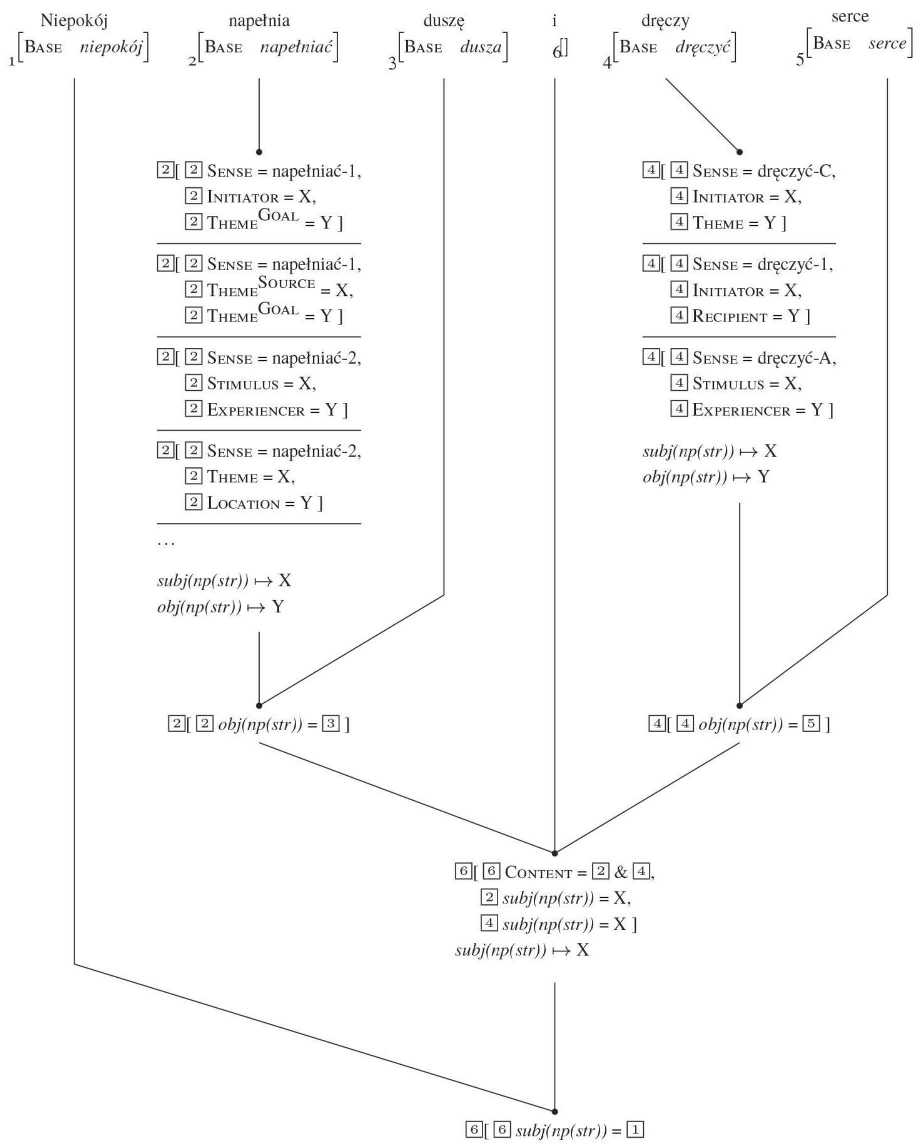
tactic arguments will be present in the utterance, we can create whole structure on spot with gaps to be filled by structures representing aforementioned arguments. To be able to fill corresponding argument positions, those will be given „aliases" that can be used outside of node where the structure is created. Each of them corresponds to a variable that is shared among all alternative predicate-argument structures. Then we need to assign a predicate-arguments structure of an argument to an „alias", which should be interpreted as filling corresponding attribute value in the chosen structure.

Similar procedure may be used to pack different senses in non-disambiguated forests. We do not create Sense attribute in leaf nodes, as we do not know the sense yet. Instead we add Base attibute, which represents the lemma of corresponding word (without its sense being disabiguated). In predicate-argument nodes corresponding to syntactic nodes where list of arguments changes appears, we create alternatives with all corresponding senses and semantic interpretations of arguments. As all those alternatives are syntactically identical (they come from the same node from syntactic forest), either of them can be actually used for creation of further nodes. A non-disambiguated predicate-argument forest packed with this procedure can be seen in Fig. 5.

In this forest all leaves have only their lemmas assigned. For verbs we do know syntactic structure of dependants. Both *napełniać* '*fill*' and *dręczyć* '*torment*' have two dependants. One of them is a nominal subject (*subj(np(str))*) and the other is an nominal object[1] (*obj(np(str))*). While this information reduces possible interpretation of those verbs, they are still numerous. Let us look at three possible interpretations for verb *dręczyć* '*torment*' presented in Fig. 5. Going from top to bottom:

- *dręczyć-C* '*mistreat/abuse*' – *Janek dręczy Anię*

[1] An object in *Walenty* is something that becomes a subject in passive voice.



Niepokój
napełnia
duszę
i
dręczy
serce
1[BASE niepokój]
2[BASE napełniać]
3[BASE dusza]
6[]
4[BASE dręczyć]
5[BASE serce]
2[2 SENSE = napełniać-1, 2 INITIATOR = X, 2 THEME^GOAL = Y]
2[2 SENSE = napełniać-1, 2 THEME^SOURCE = X, 2 THEME^GOAL = Y]
2[2 SENSE = napełniać-2, 2 STIMULUS = X, 2 EXPERIENCER = Y]
2[2 SENSE = napełniać-2, 2 THEME = X, 2 LOCATION = Y]
...
subj(np(str)) ↦ X
obj(np(str)) ↦ Y
4[4 SENSE = dręczyć-C, 4 INITIATOR = X, 4 THEME = Y]
4[4 SENSE = dręczyć-1, 4 INITIATOR = X, 4 RECIPIENT = Y]
4[4 SENSE = dręczyć-A, 4 STIMULUS = X, 4 EXPERIENCER = Y]
subj(np(str)) ↦ X
obj(np(str)) ↦ Y
2[2 obj(np(str)) = 3]
4[4 obj(np(str)) = 5]
6[6 CONTENT = 2 & 4, 2 subj(np(str)) = X, 4 subj(np(str)) = X]
subj(np(str)) ↦ X
6[6 subj(np(str)) = 1

Figure 5: Packed non-disambiguated predicate-argument forest.

(szczypaniem). ‘*John is abusing Anna (pinching her)*’;

- *dręczyć-1* ‘*bother*’ – *Rodzice dręczyli Janka (pytaniami).* ‘*Parents were bothering John (asking questions).*’;

- *dręczyć-2* ‘*worry*’ – *Coś mnie dręczy.* ‘*Something worries me*’.

Those meanings, while separate, can be expressed with the same syntactic structure. As those are indistinguishable based only on syntactic features, those meanings are packed into a single node with an interface (in form of mapping) that allows to ignore their semantic differences (e.g., different role assigned to arguments represented by same syntactic phrase type).

6. Reading out a single predicate-argument tree

Getting a single tree from a shared forest is a straightforward procedure. First you go from the root to leaves selecting a single option from all alternatives. Then, as you go bottom-up, you perform operation described on selected option and store node created this way. The structure you get in the root is one of predicate-argument structures for the utterance.

To get a predicate-argument tree for a chosen syntactic tree, you choose only from alternatives available for that tree – you can only choose a particular option, only if all children that were used to create it correspond to nodes in the given syntactic tree.

7. Complexity

Time and space complexity of a predicate-argument forest packed in the way described in this paper depends mainly on space complexity of syntactic forest that is used. Easiest way to analyze it is looking at differences in chart-parser tables used to create both.

First, let us look at the size of data stored in each cell. For most syntactic parsers number of entries in them is bound by a constant. The same thing isn't true for predicate-argument parser. The multiplication comes from two sources. As can be seen in Fig. 5, the cell can have multiple semantic interpretations. This factor is actually bound by a constant that can be derived from valence dictionary – maximal number of semantic interpretations for a single schema. The other comes from the fact that a single phrase can have multiple (lexical) semantic centers. That number can only be bound by the number of words in that phrase (in case of e.g., a genitive cluster). Altogether, there can be $O(n)$ times more data in each cell of chart-parser.

This means that space complexity of shared forest is $O(n)$ times larger then that of original syntactic forest and time complexity is $O(n^2)$ times larger (in case of binary grammar, in general it's $O(n^k)$ times larger, where k is maximal length of a rule). It is worth noting that increase occurs in some rare situations and in most cases both time and space complexity are in the same order of magnitude as for original syntactic parser.

If the grammar used to create predicate-argument forest (and original syntactic forest) is context-free (i.e., if a structure is connected to a syntactic node, then it can be used in any of productions that use that node), time complexity can be further decreased. If you create a node with all options connected to a single syntactic node and create outside name for that alternative that can be used by further nodes, you will manage to reduce number of „output symbols" (symbols going to other nodes in chart-parser) to a constant. This way time complexity of predicate-argument forest creation will be exactly in the same order of magnitude as size of syntactic forest.

8. Conclusions and future work

This paper shows a sharing algorithm that can create predicate-argument forest out of syntactic forest without large increase in time and space complexity with respect to size of the original forest. Such parser for Polish is a base for construction of semantic treebank for Polish.

In the future we plan to extend our grammar so it would add some semantic information that is not strictly involved in predicate-argument structure (e.g., generalized quantifiers). We will also include information about preferred argument content (in form of selectional preferences from *Walenty*) to further decrease size of forests by removing semantically incorrect structures.

9. References

Hajnicz, Elżbieta, Anna Andrzejczuk, and Tomasz Bartosiak, 2016a. Semantic Layer of the Valence Dictionary of Polish *Walenty*. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, and Stelios Piperidis (eds.), *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC-2016)*. Portorož, Slovenia: ELRA.

Hajnicz, Elżbieta, Agnieszka Patejuk, Adam Przepiórkowski, and Marcin Woliński, 2016b. *Walenty*: słownik walencyjny języka polskiego z bogatym komponentem frazeologicznym. In Karolína Skwarska and Elżbieta Kaczmarska (eds.), *Výzkum slovesné valence ve slovanských zemích*. Prague, Czech Republic: Slovanský ústav Akademie věd ČR, pages 71–102.

Przepiórkowski, Adam, Elżbieta Hajnicz, Anna Andrzejczuk, Agnieszka Patejuk, and Marcin Woliński, 2017. Walenty: gruntowny składniowo-semantyczny słownik walencyjny języka polskiego. *Język Polski*, XCVII(1):30–47.

Przepiórkowski, Adam, Elżbieta Hajnicz, Agnieszka Patejuk, Filip Skwarski, Marcin Woliński, and Marek Świdziński, 2014. Walenty: Towards a comprehensive valence dictionary of Polish. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (eds.), *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014)*. Reykjavík, Iceland: ELRA.

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

Świdziński, Marek and Marcin Woliński, 2010. Towards a bank of constituent parse trees for Polish. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala (eds.), *Proceedings of the International Conference on Text, Speech and Dialogue TSD 2010*, volume 6231 of *LNAI*. Brno, Czech Republic: Springer-Verlag.

Woliński, Marcin, 2004. *Komputerowa weryfikacja gramatyki Świdzińskiego*. PhD thesis, Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland.

Woliński, Marcin, 2015. Deploying the new valency dictionary Walenty in a DCG parser of Polish. In Markus Dickinson, Erhard Hinrichs, Agnieszka Patejuk, and Adam Przepiórkowski (eds.), *Proceedings of the 14th Workshop on Treebanks and Linguistic Theories*. Warsaw, Poland: ICS PAS.