MTT-compatible computationally effective surface-syntactic parser

Tomasz Obrebski
Technical University of Poznań*
Adam Mickiewicz University
Poznań, Poland
Tomasz.Obrebski@put.poznan.pl

Mots-clefs – Keywords

analyse en grammaire de dépendances, efficacité, syntaxe de surface
dependency parsing, efficiency, surface syntax

Abstract - Resume

The paper presents a dependency parser developed for Polish. Our primary concern was computational efficiency. The representation of syntactic structure and the formulation of syntactic description is close to that used in the surface-syntactic component of the MTT. We describe the way the syntactic description is specified and outline the parsing process. The parser uses syntactic dependency graph to represent an ambiguous syntactic structure. The comparison to the MTT and to related works on dependency parsing is included.

Nous présentons un analyseur syntaxique développé pour le polonais. La représentation de la structure syntaxique et la formulation de la description syntaxique est proche de celle qui est utilisée dans le module de syntaxe de surface de la TST. Nous décrivons la façon dont la description syntaxique est formulée ainsi que le déroulement de l'analyse. La comparaison avec la TST et avec d'autres travaux sur l'analyse en grammaires de dépendances est incluse.

* This work was partially financed by Poznań University of Technology, Research Project No 45-083/03/DS.

1 Introduction

The paper presents a dependency parser called **dgp** (**d**ependency **g**raph **p**arser), developed as the result of the author's research on computationally efficient method of dependency parsing well suited for Polish. More precisely, our goal was to develop a parsing technique which could be applied in efficient tools for raw text corpora processing. The time efficiency was our primary concern and we did not aim at achieving the largest possible coverage of syntactic constructions. In the first step, which, in our opinion, was accomplished successfully, we excluded from the language subset under consideration the discontinuous constructions and ellipsis.

The way the syntactic structure is represented and the way the syntactic description is formulated were inspired mainly by: the surface-syntactic component of the MTT (Mel'čuk, 1988; Mel'čuk & Pertsov, 1988), in what concerns dependency description of natural language syntax, and the Saloni and Świdziński's *Syntax of contemporary Polish* (Saloni & Świdziński, 1998) in what concerns the Polish syntax.

2 Syntactic structure representation

The syntactic structure is represented as a dependency tree with arcs labelled with dependency types. Additionally, so called syntactic flags may be assigned to nodes. A syntactic flag corresponds to a purely syntactic attribute in Saloni and Świdziński's terminology and reflects a property attributed to a word due to the syntactic context it appears in. Flags are assigned to nodes during syntactic analysis. For example, the REL flag is used to indicate the property of being a head of a phrase containing a relative pronoun. A verb with the REL flag has different connectivity than a verb without it. In Figure 1 an example of dependency tree is shown.

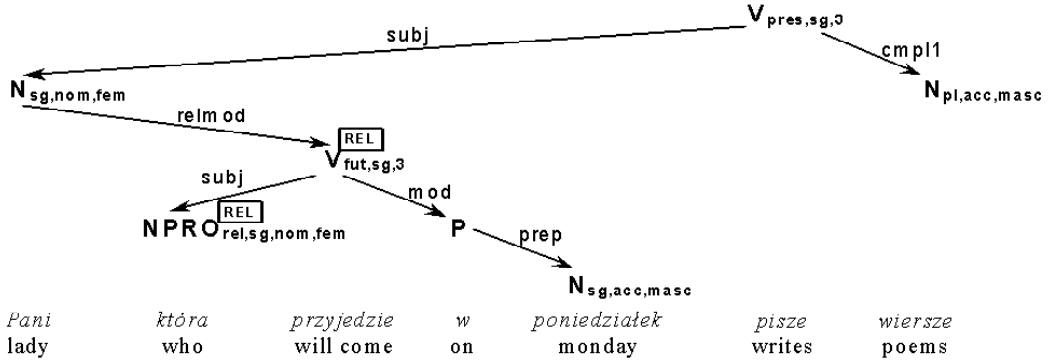

Figure 1: Dependency tree

3 Syntactic description

The syntactic description of a language includes the following components (all of them are parser's input data):

- declarations of parts of speech
- declarations of morpho-syntactic attributes and their values

- declarations of syntactic flags
- definitions of distribution classes
- definitions of agreement types
- definitions of dependency types
- dependency rules
- flag propagation rules
- long distance agreement rules

We will shortly comment on each of the above points. Syntactically relevant properties of a word are expressed by assigning to it a syntactic category: the part of speech and values for relevant morpho-syntactic attributes (eg. number, gender, aspect, conjunction type, etc.). Attribute values are atomic symbols or finite sets thereof. Examples: $\mathtt{N}_{\mathtt{sg,nom,fem}}$, $\mathtt{V}_{\mathtt{pl,3,neut}}$.

By a grammatical class we call a set of words which share certain syntactically relevant properties. Classes defined by imposing restrictions on the part of speech and attribute values are called morpho-syntactic classes. For example N denotes the class of all nouns, $*_{\mathtt{acc}}$ – the class of all words in accusative (the asterisk stands for 'any part of speech').

We also introduce the notions of passive and active distribution class (closely related to the notions of passive and active valency). The passive (active) distribution class is a class of words exhibiting common properties relative to their connectivity as dependents (heads). The words belonging to a given passive distribution class have, in MTT terms, the same passive valency (at least in certain syntactic contexts). For example, we can define the distribution class of words passively equivalent to nouns as $\mathtt{N}^{-} = \mathtt{N} \cup \mathtt{NPRO}$, where the superscript 'minus' indicates passive ('plus' is used for active), NPRO stands for nominal pronoun. The reason for the introduction of distribution classes was to enable generalizations in syntactic descriptions.

An agreement type is – in our technical sense – a binary relation among syntactic categories defined usually (but not necessarily) in terms of equality or unifiability of certain attribute values. Agreement types are used for implementing the phenomena of agreement and congruence (in the sense of (Mel'čuk, to appear)). We also use specific agreement types to handle coordination by specifying, which classes of words may be conjoined. The predefined agreement types are those relying on unifiability of single attribute values.

Dependency type definitions introduce dependency type names and assign them the following properties:

1. dependency status: facultative, single (non-repeatable) or obligatory (for words belonging to certain categories);
2. relative position of the dependent with respect to the head: left, right or free-order;
3. relative position of the dependent with respect to other dependents and the head: initial, head-adjacent, final;
4. agreement types required between the head and the dependent;
5. systemic or lexical government: systemic government is specified by determining the grammatical class to which the dependent must belong; lexical government is specified by determining, which connotational requirement the given dependency type is related to (see below);

For example the following dependency type definition:

TYPE subj = SGL, FREE, AGR(Gender,Number), GOV($*_{\mathtt{nom}}$).

defines the type subj as having the following properties: single, free-order, agreement with respect to gender and number, systemic government of the nominative case.

Dependency rules determine, for each dependency type, which pairs of words may be connected by a dependency of this type. Dependency rule application may be conditioned by the presence or absence of a certain flag. For example, the rule

RULE relmod: $\mathtt{N} \rightarrow \mathtt{V}_{\mathtt{fin}}$ //REL.

reads that a dependency of type rel may by used to connect a noun with a finite verb possessing the REL flag. Constraints of the same kind as those appearing in dependency type definitions may also be specified in dependency rules (cf. rule (5) below).

Flag propagation rules describe the principles of flag assignment. We distinguish flag initialisation rules and flag transmission rules. The following rules (the backslash is the set difference operator):

INIT REL $\mathtt{NPRO}_{\mathtt{rel}}$.
UP REL $*$: $* \rightarrow * \setminus \mathtt{V}_{\mathtt{fin}}$.

can be read as, respectively: 'assign the flag REL to each relative pronoun' and 'transmit the flag REL upwards, i.e. from the dependent to the head, through a dependency of any type, if the dependent is not a finite verb' (i.e. 'transmit the flag REL upwards to the nearest finite verb').

Long distance agreement rules are used to impose requirements of agreement of a specific type on nodes which are not directly connected by a dependency. They are formulated by specifying the type of agreement, the grammatical class of the source node, the description of the path to the destination node, and the grammatical class of the destination node. For example the rule

AGR(Number,Gender) : $\mathtt{NPRO}_{\mathtt{rel}} \ll$relmod N

expresses the requirement of agreement with respect to number and gender between a relative pronoun and the noun to which the relative clause containing this pronoun is attached. The expression $\ll$relmod describes any path leading upwards (to a transitive head) ending with a dependency of type relmod and not containing other dependencies of type relmod. A rule

AGR(COORD) : $*$ >coord CONJ >conj $*$

can be used to handle coordination, provided that the agreement type COORD determines pairs of grammatical categories which may be conjoined. The path is described as directed downwards consisting of a dependency of type coord, a conjunction, and a dependency of type conj.

The following grammar sample contains definitions and rules which allow to analyse the expression *pani, która przyjedzie, pisze* in the way shown in Figure 1.

(1) CLASS $\mathtt{N}^{-} = \mathtt{N} \cup \mathtt{NPRO}$.
(2) TYPE subj = FREE, SGL, AGR(Nunmer,Gender), GOV($*_{\mathtt{nom}}$).
(3) TYPE relmod = RIGHT, SGL.
(4) RULE subj: $\mathtt{V}_{\mathtt{fin}} \rightarrow \mathtt{N}^{-} \setminus \mathtt{NPRO}_{\mathtt{rel}}$.
(5) RULE subj+INIT: $\mathtt{V}_{\mathtt{fin}} \rightarrow \mathtt{NPRO}_{\mathtt{rel}}$.
(6) RULE relmod: $\mathtt{N} \rightarrow \mathtt{V}_{\mathtt{fin}} // \mathtt{REL}$
(7) AGR(Number,Gender): $\mathtt{NPRO}_{\mathtt{rel}} <<$relmod N.
(8) INIT REL $\mathtt{NPRO}_{\mathtt{rel}}$.
(9) UP REL $*$: $* \rightarrow * \setminus \mathtt{V}_{\mathtt{fin}}$.

Most of the statements were already explained. In the line (3) the `relmod` type is defined: the dependent must follow the head, the dependency is non-repeatable. Line (4) contains the main rule for `subj` type, used to connect *pisze* with *pani*. This rule does not cover subjects which are relative pronouns. Subjects of this type are handled by rule (5), introducing additional constraint (`INIT`) which requires the initial position of the relative pronoun in the verb's phrase.

The dictionary used by the parser is a dictionary of word-forms. It provides the following information about a word-form: the lexeme identifier, the grammatical category, the description of connotational properties (optional).

Connotational properties may include information on the grammatical classes required for the head and up to three dependents. Dependency types supposed to link the word with the connotated words are not specified here. The example below shows the interaction between the dictionary and the grammar in this respect.

dał;dać, $\mathtt{V}_{\mathtt{past,sg,3,masc}}(\mathtt{N}^{-}_{\mathtt{acc}})(\mathtt{N}^{-}_{\mathtt{dat}})$

```
TYPE cmpl1 = FREE, CON#1.        RULE cmpl1:  V → *.
TYPE cmpl2 = FREE, CON#2.        RULE cmpl2:  V → *.
```

The dictionary entry specifies two connotational requirements concerning dependents (the expressions in brackets specify the grammatical classes they must belong to). The first specifies a class of words passively equivalent to nouns in accusative. The second – a similar class, but in dative. The definition of dependency type `cmpl1` reads that it is used to connect a word with its first connotational requirement (`CON#1`). Finally, the dependency rule for `cmpl1` reads only that a dependency of this type connects a verb with its dependent; nothing is said here about the dependent's category.

4 Comparison to MTT

What we called above a syntactic description corresponds to the surfice-syntactic component of the MTT. Dependency rules correspond to syntagms, with the difference that all the constraints common to all syntagms for a given dependency type are factorized and attached directly to the type in its definition. Only rule-specific constraints are placed in dependency rules (cf. rule (5) in Section 3). Constraints refer either to the information from the morphological level (word order, morphological dependencies: government, agreement, congruence), included in morphological structures within MTT, or express the syntactic structure well-formedness conditions (eg. non-repeatability, obligatoriness).

Contrary to syntagms, we do not allow dependency rules to refer to nodes other than the two directly connected by the dependency being described. Obligatoriness conditions and flag propagation mechanism are used instead. For example, making the dependency connecting a conjunction with the second conjunt obligatory for the conjunction garanties that the conjunction is always connected with both conjuncts: with the left one – because the conjunction must have a head, and with the right one – because this dependency is obligatory. Instead of requiring from the verbal head of a relative clause to have a relative pronoun as a transitive dependent (cf. the syntagm for the relative clause in (Mel'čuk & Pertsov, 1988, s. 395)), it is checked for the presence of the `REL` flag, which is transmitted from the pronoun up to the verb.

5 Parser overview

5.1 Parsing steps

The ambiguous morphological analysis output is represented in the form of a directed acyclic graph, called lexical graph. The nodes of this graph represent syntactic categories corresponding to possible interpretations of word-forms of the input sentence. Arcs represent the direct successor relation. As an example, let us consider the following sentence:

Kobiety	*przyniosły*	*k*ozie	*mleko*	*w*	*butelce.*
women	brought	goat (N)	milk	in	bottle
		goat (ADJ)			

The word-form *kozie* may be interpreted as the adjective 'goat', modifying the noun *mleko*, or as the noun 'goat' in dative, the complement to *przyniosły*. The whole sentence can be read as 'Women brought goat milk in a bottle' or 'Women brought milk in a bottle for the goat' with an additional ambiguous attachment of the prepositional phrase *w butelce*. Figure 2 shows the lexical graph for the sentence (not all word-form interpretations are included). The parsing

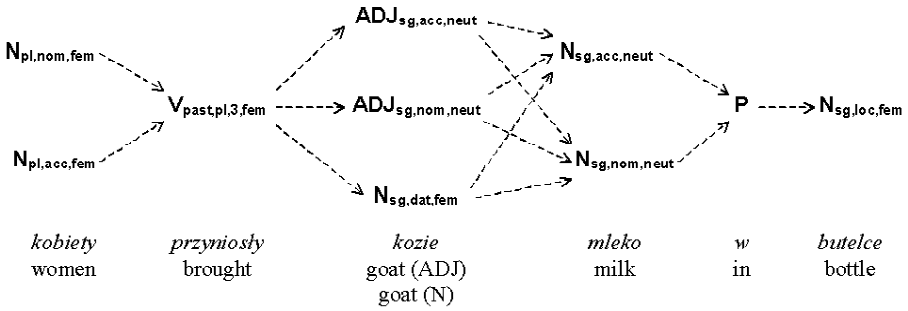

Figure 2: Lexical graph

process proceeds in two phases: the dependency graph construction and the dependency tree generation. The dependency graph is an ambiguous representation of the syntactic structure. The set of nodes of the dependency graph is basically (see Subsection 5.2) the same as the set of nodes of the lexical graph. The arcs represent possible dependencies between words. The most important properties of the dependency graph are:

- P1 The graph is projective[1]. As a consequence, if an arc from w_i to w_j in present in the graph, the graph has a sub-graph which is a projective dependency tree built on all nodes of some lexical path between w_i and w_j.
- P2 All dependency trees that can be constructed for a sentence, taking into account all possible interpretations of word-forms, are sub-graphs of the graph.

In Figure 3 the dependency graph for the example sentence is shown. (The numbers below arcs do not belong to the graph.)

The second step is the generation of dependency trees. In Figure 3 the numbers 1,2,3, and 4 indicate the dependencies belonging to the four dependency trees which can be generated from the graph.

[1]The projectivity property is defined in the same way as for dependency trees.

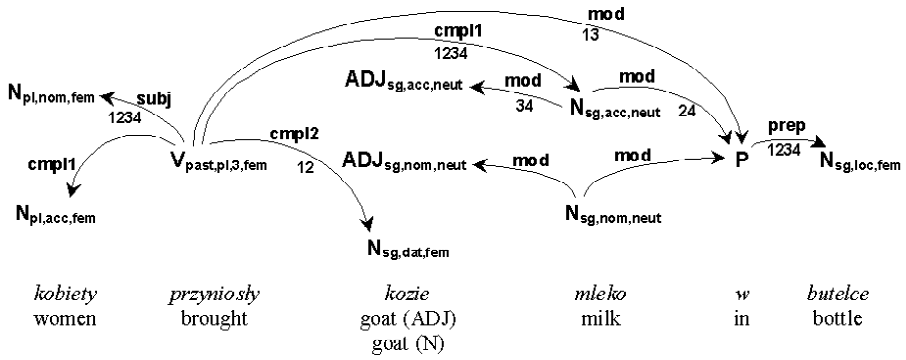

Figure 3: Dependency graph

5.2 Flag propagation and long distance agreement handling

When adding a dependency arc to the graph results in assigning a flag to one of the nodes (as the result of application of a flag propagation rule), we cannot simply set the flag at this node. This is because each node can be shared between many different syntactic interpretations. Consider the following example:

pani,	*która*	*przyjedzie*
lady	who	will come

The pronoun *która* may be interpreted as a relative or interrogative pronoun. The connectivity (passive valency) of the verb connected to a relative pronoun and to an interrogative one differ from one another and also from the connectivity of the verb not connected to any of them. Therefore, each time a flag is to be assigned to a node, we must create its duplicate and assign the flag to this duplicate, leaving the original node unchanged.[2] In our example we obtain three verb nodes: the first, connected to the relative pronoun, with the REL flag assigned, the second, connected to the interrogative pronoun, with the INT flag assigned, and the third, the original node, not connected to any pronoun with no flag assigned (see Figure 4).

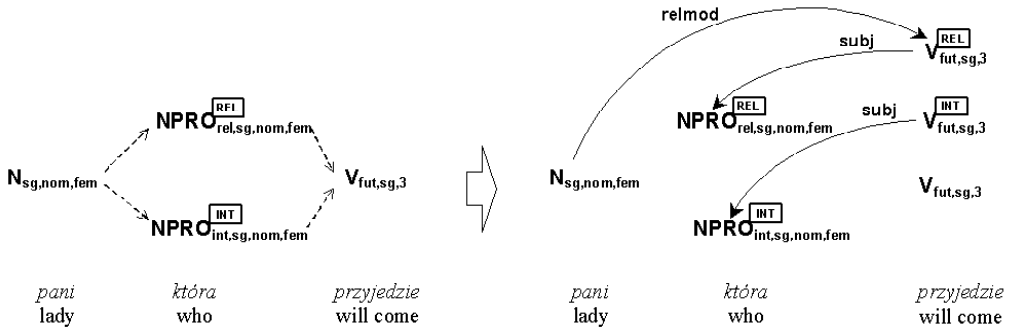

Figure 4: Node duplication

For handling long distance agreement a similar mechanism is used. The information package passed from node to node includes the following information: the description of the path remaining to traverse, the grammatical class of the destination node, the agreement type, and the

[2]This is in some way the reverse of the structure sharing technique used for compacting the representation of tree forests. Instead of considering which elements of two or more alternative structures can be shared we consider which elements have to be separated. The first technique requires performing a search, the second does not.

agreement value of the source node. When the last dependency arc specified in the path description is about to be established, the agreement condition is checked. If the check fails, the arc is not added to the graph.

5.3 Algorithm

In this section the core of the graph construction algorithm is outlined. The graph is computed incrementally from left to right. For each node w_j the following three sets are computed: the set of transitive left dependents of w_j, the set of transitive left heads of w_j, and the set of nodes visible to w_j on its left[3]. The algorithm tries to connect the current node w_j with all preceding nodes w_i, considered in descending order. An arcs can only be added if w_i is visible to w_j. The set of nodes visible to w_j on the left is initialized with transitive left heads of left neighbours of w_j. After having established a leftward dependency connecting w_j to w_i, the transitive left dependents of w_i become transitive left dependents of w_j and the nodes visible to w_i on the left become visible on the left to w_j. After having established a rightward dependency, the set of transitive left heads is updated in the same way.

The theoretical complexity of the algorithm is $O(n^3)$, where n is the number of nodes in the graph, which is proportional to the length of the input sentence, assuming that the number of alternative interpretations of one word-form is bounded. In the implementation the complexity is reduced[4] to $O(n^2)$. The mechanisms involving node duplication does not change the complexity with respect to the length of the sentence, under the restriction that flags and agreement information may be propagated only from left to right. The number of possible different duplicates does not depend on the number of nodes. This restriction does not decrease the language subset considerably from the practical point of view. Syntactic constructions whose analysis requires flags to be propagated in the backward direction are relatively infrequent (eg. relative clauses with the relative pronoun deeply embeded into the structure[5], of the type 'the law for the rejection of which you voted').

After having constructed the graph, we have, apart from the graph itself, the information on each node's all (possible) transitive left heads and dependents, as well as nodes visible on the left. Using this information, all trees can be generated in time $O(dm^2)$ ($O(dm)$ in the implementation), where d – number of trees, m – number of nodes in a tree. Detailed description of the algorithm, and the way the constraints are handled, may be found in (Obrebski, 2003) and (Obrebski, 2002).

[3]The node w_i is a transitive left dependent (head) of w_j, if w_i is a transitive dependent (head) of w_j and all intervening dependencies are directed leftwards (rightwards). The node w_i is visible to w_j (on the left), if w_i is a transitive left head of a left neighbour of a transitive left dependent of w_j ('neighbour' refers to the linear order). The visibility relation expresses the structural condition for two nodes to be connectable, preserving the projecticity property.

[4]In order to add eg. the set of transitive left heads of one node to the set of transitive left heads of another node the union of two ordered sets has to be computed. This operation has linear complexity in theory. In the calculation of the implementational complexity, however, it is considered as a constant-time operation, as it is realized as the sum of bit vectors of constant size.

[5]Cases where the relative pronoun in preceded by a preposition, which are frequent, are handled using a simple 'hacking' technique.

5.4 Comparison to related works

The syntactic description is formulated in a constraint-based manner and falls into the category of constraint-based dependency grammars (eg. (Maruyama, 1990), (Duchier, 1999)), as opposed to Hays-style formulations of dependency grammar (Hays, 1964; Lombardo & Lesmo, 1998). Many similarities can be seen with Duchier's way of grammar specification. The parsing procedure, however, has more in common with all-paths tabular (item-based) algorithms, particularly with the Eisner's one, using the concept of 'span' (Eisner, 1996). The information computed in our algorithm roughly corresponds to storing all possible 'leftward–', 'rightward–', and 'no-covering-link spans'[6] in Eisner's terminology. The difference is that while adding eg. a leftward dependency a 'no-covering-link span' is combined with all adjacent 'leftward spans' at the same time, and is combined with all adjacent 'no-covering-link spans' at the same time, by a set union operation. The theoretical complexity of both algorithms is $O(n^3)$, but the reduction to $O(n^2)$ in the implementation is not possible in Eisner's algorithm.

The use of dependency graph to represent ambiguous dependency structure has already been proposed by Barbero and Lombardo (Barbero & Lombardo, 1995). The main difference is that the dependency graph which we use does not contain explicit information about the parse trees[7]. We do not know which arcs can eventualy appear together in the trees. In order to be able to construct the trees efficiently we need additional information for ech node: about its transitive left heads and dependents and nodes visible on the left.

6 Test results

The parser has been tested on the set of 170 sentences from the test-set for syntactic analysers of Polish.[8] These were all correct sentences belonging to categories 'basic' and 'complicated' (the sentences from the category 'peripheral' were omitted). 156 sentences were parsed correctly.[9] The sentences which were not parsed contained ellipsis, discontinuous expressions or complex subjects of the type *Jan i Maria* (John and Mary). In the letter case agreement takes place between the verb and the coordinated structure as a whole, which has the plural number, while the head of the structure is in singular.

The time of dependency graph construction (measured on PC 1,2 GHz) is around 2ms for a sentence of length 10 (word-forms), 5ms for the sentence of length 20, and 10ms for a sentence of length 40. The time of generation of one tree is about 0,1ms.

[6] Leftward and rightward spans are projective subtrees in which, respectively, only the righmost or the leftmost node has no head. They correspond to subtrees between a node and its transitive left dependent and transitive left head, respectively. A 'no-covering-link span' is a projective subgraph in which only the leftmost and the righmost node have no head. It corresponds to a subgraph between two nodes visible one to another.

[7] After constructing the graph we do not even know if there exists any tree in the graph. This can be easily computed, hovewer, by checking if there exists a node which is a transitive left head of a final node (with no successor in the lexical graph) and such that an initial node is its transitive left dependent.

[8] The test-set was constructed in the Institute of Computer Science of the Polish Academy of Sciences and is available from: http://www.ipipan.waw.pl/ agn/sentences.htm.

[9] 'Parsed correctly' means that none of the output parses could be considered as incorrect, taking into account the limited morpho-syntactic information about words, available to the parser.

7 Applications

Considering the restricted syntactic coverage, the **dgp** parser may be interesting for the application fields where efficiency considerations are crucial while incomplete syntactic analysis is acceptable. In particular, we mean here the area of large corpora processing, including location of phrases of a specific type, syntactic concordancing, and syntactic and lexical information retrieval. It is worth to note that in all these applications the computations can be carried directly on the graph, without the necessity of generating trees.

The application of the parser we are particularly interested in is syntactic pattern location in raw text corpora, especially in the context of semi-automatic syntactic dictionary data retrieval. Such a tool is currently developed.

References

BARBERO C. & LOMBARDO V. (1995). Dependency graphs in natural language processing. In *Lecture Notes in Artificial Intelligence, 992*, p. 115–126. Springer-Verlag.

DUCHIER D. (1999). Axiomatizing dependency parsing using set constraints. In *Proc. 6th Meeting of Mathematics of Language*, Orlando.

EISNER J. M. (1996). Three new probabilistic models of dependency parsing: An exploration. In *Proceedings of COLING-96*, Copenhagen.

HAYS D. (1964). Dependency theory: A formalism and some observations. *Language*, **40**(4), 511–525.

LOMBARDO V. & LESMO L. (1998). Formal aspects and parsing issues of dependency theory. In *Proceedings of COLING/ACL'98*, p. 787–93, Montréal.

MARUYAMA H. (1990). Structural disambiguation with constraint propagation. In *Proceedings of the 28th ACL*, p. 31–38, Pittsburgh, PA.

MEL'ČUK I. A. (1988). *Dependency Syntax: Theory and Practice*. State University of New York Press.

MEL'ČUK I. A. (to appear). Dependency in linguistic description.

MEL'ČUK I. A. & PERTSOV N. V. (1988). *Surface syntax of English. A formal Model within the Meaning-Text framework.* Amsterdam: Beniamins.

OBREBSKI T. (2002). *Automatyczna analiza składniowa jezyka polskiego z wykorzystaniem gramatyki zależnościowej (Automatic syntactic analysis of Polish with the use of dependency grammar).* PhD dissertation, Polish Academy of Sciences, Institute of Computer Science.

OBREBSKI T. (2003). *Dependency parsing using dependency graph for storing ambiguous structure*. Technical report, Poznań University of Technology, Institute of Control and Information Engineering.

SALONI Z. & ŚWIDZIŃSKI M. (1998). *Składnia współczesnego jezyka polskiego (Syntax of contemporary Polish).* Warszawa: Wydawnictwo Naukowe PWN, fourth edition.