Platform for Full-Syntax Grammar Development Using Meta-grammar Constructs

Aleš Horák and Vladimír Kadlec

Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno, Czech Republic
{hales,xkadlec}@fi.muni.cz

Abstract. This paper describes a combination of tools necessary for full or deep syntactic parsing of natural language – the syntactic parser synt, the graphical Grammar Development Workbench, GDW and the VerbaLex verb valency lexicon tools.
We describe the development of the mentioned tools and how they integrate into one system that allows a team of experts (computational linguists as well as programmers) to cooperate on the development of grammar covering all Czech language phenomena.

1 Introduction

Full parsing of natural language sentences is a complicated task that provides a transition from unstructured text into structural information suitable for all forms of information retrieval. Any kind of higher level language understanding and/or semantic processing must rely on the results of syntactic parsing. The quality of state-of-the-art syntactic parsers [1, 2] is still not completely satisfiable even for analytical languages like English. In case of free-word-order languages like Czech, the situation is even more complicated.

At the Centre for Natural Language Processing at the Faculty of Informatics, Masaryk University in Brno, a full syntactic parsing system is being developed since 1997. The core of the system, the syntactic parser `synt` is based on the meta-grammar formalism (e.g. [3–5]), which allows to specify complex sentence constituent combination rules in a maintainable way. The composition of the formalism resembles structures in Lexical functional grammars [6] – the meta-grammar rules are expanded within the guidance of combinatorial constructs plus contextual constraints and are supplemented with additional actions and agreement tests.

The described full grammar development platform consists of the following components:

- the Czech morphological analyser `ajka` [7]. This tagger provides non-disambiguated output of plain text words. The `ajka` tagger covers almost 400.000 Czech word lemmas, which generates over 6 million word forms.
- the VerbaLex verb valency lexicon tools [8]. The contextual constraints that safeguard the syntactic analysis of free-word order language need extra lexical-semantic

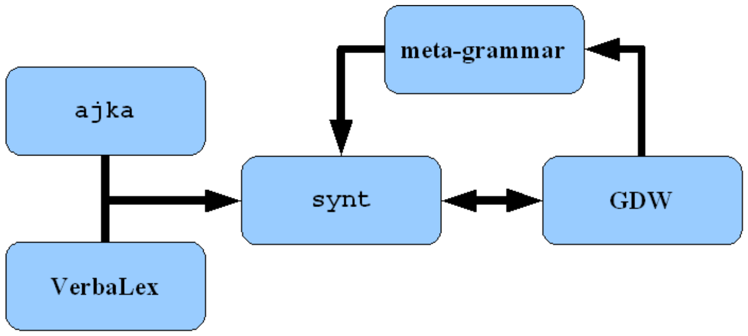


Fig. 1. Interaction of our tools

information about the sentence constituents. The creation of new verb valency lexicon which contains so called *complex valency frames* has thus become a part of the grammar development platform.
- the deep syntactic parser `synt`, see [9–11]. The parser development concentrates on high coverage on general corpus sentences ($> 90\,\%$). The parser is able to work with several parsing algorithms (GLR parsing, top-down, bottom-up and head-driven chart parsing). The `synt` parser also introduced a new variant of the head-driven technique, the head-driven dependent dot move [12] algorithm. This parsing technique allows to parse natural language sentences within median time of less than 0.1s/sentence.
- the user interface for linguists – the Grammar Development Workbench, GDW [13]. The parser `synt` provides a command line interface, which is suitable for all forms of batch processing. However, the grammar development conducted by linguistic experts combines working with `synt` input parameters (the meta-grammar, the corpus text, parameters guiding the analysis) as well as thorough studying of `synt` outputs (tagged sentences, syntactic or dependency trees, chart graphs). All these tasks can be solved via GDW.

In the following sections, we describe the functionality of the parsing platform in more detailed way and demonstrate the cooperation of its components on several examples.

2 The System Architecture

The grammar development platform includes several tools as described above. The Figure 1 shows, how these tools interact with each other. The Czech morphological analyser `ajka` and the VerbaLex verb valency lexicon provide lexical information about words or multi-word expressions in an input sentence. The syntactic parser `synt` uses the data from lexicon and the meta-grammar to create syntactic structures. The Grammar Development Workbench (GDW) is a user interface able to control the parser. Main tasks of linguistic experts working with GDW include running the parser, studying its outputs, edit and test the meta-grammar and also build a tree bank of correct derivation trees used for probability estimations and testing of grammar changes.

3 VerbaLex

The lexicon of verb valencies, VerbaLex [14], was created in 2005. VerbaLex is based on three valuable language resources for Czech, three independent electronic dictionaries of verb valency frames.

The first resource, Czech WordNet valency frames dictionary, was created during the Balkanet project and contains semantic roles and links to the Czech WordNet semantic network. The other resource, VALLEX 1.0 [15], is a lexicon based on the formalism of the Functional Generative Description (FGD) and was developed during the Prague Dependency Treebank (PDT) project [16]. The third source of information for VerbaLex is the syntactic lexicon of verb valencies denoted as BRIEF, which originated at FI MU Brno in 1996 [17].

The resulting lexicon VerbaLex comprehends all the information found in these resources plus additional relevant information such as verb aspect, verb synonymy, types of use and semantic verb classes based on results of the VerbNet project [18]. The information in VerbaLex is organized in the form of *complex valency frames* (CVF). All the valency information in VerbaLex is specified regarding the particular verb senses, not only the verb lemmata, as it was found in some of the sources. The current work on the lexicon data aims at enlarging the lexicon to the size of about 16.000 Czech verbs. The VerbaLex lexicon displays syntactic dependencies of sentence constituents, their semantic roles and links to the corresponding Czech WordNet classes.

An example of such verb frame is presented in the Figure 2.

4 Grammar

The meta-grammar concept in the `synt` system [11] consists of three grammar forms denoted as G1, G2 and G3. Human experts work with the meta-grammar form, which encompasses high-level generative constructs reflecting the meta-level natural language phenomena like word order constraints, and enables to describe the language with a maintainable number of rules. The meta-grammar serves as a base for the second grammar form which comes into existence by expanding the constructs. This grammar consists of context-free rules equipped with feature agreement tests and other contextual actions. The last phase of grammar induction lies in the transformation of the tests into standard rules of the expanded grammar with the actions remaining to guarantee the contextual requirements.

Meta-grammar (G1) The meta-grammar consists of global order constraints that safeguard the succession of given terminals, special flags that impose particular restrictions to given non-terminals and terminals on the right hand side (RHS) and of constructs used to generate combinations of rule elements.

Princeton WordNet: dress:2, clothe:1, enclothe:1, garb:1, raiment:1, tog:1, garment:1, habilitate:2, fit out:2, apparel:1
definition: provide with clothes or put clothes on
VerbaLex Synset: obléci:1_{pf}, oblékat:1_{impf}, obléknout:1_{pf}, ustrojit:1_{pf}, strojit:1_{impf}
=def: provide with clothes or put clothes on
=canbepassive: yes
=meaning: I
=class: dress-41.1.1
Complex valency frames:

1. obléci:1, oblékat:1, obléknout:1
   -frame: AG<person:1>$^{\mathrm{obl}}_{\mathrm{who1}}$ VERB
   PAT<person:1>$^{\mathrm{obl}}_{\mathrm{to\_whom3}}$ ART<garment:1>$^{\mathrm{obl}}_{\mathrm{what4}}$
   -synonym: ustrojit:1, strojit:1
   -example: *maminka oblékla dítěti kabát / the mother put a coat on her child*
   -attr: use: prim, reflexivity=obj_dat, mustbeimperative=no
2. obléci:1, oblékat:1, obléknout:1, ustrojit:1, strojit:1
   -frame: AG<person:1>$^{\mathrm{obl}}_{\mathrm{who1}}$ VERB
   PAT<person:1>$^{\mathrm{obl}}_{\mathrm{whom4}}$ ART<garment:1>$^{\mathrm{obl}}_{\mathrm{in+sth2}}$
   -synonym:
   -example: *maminka oblékla dítě do kabátu / the mother dressed her child in a coat*
   -attr: use: prim, reflexivity=obj_ak, mustbeimperative=no

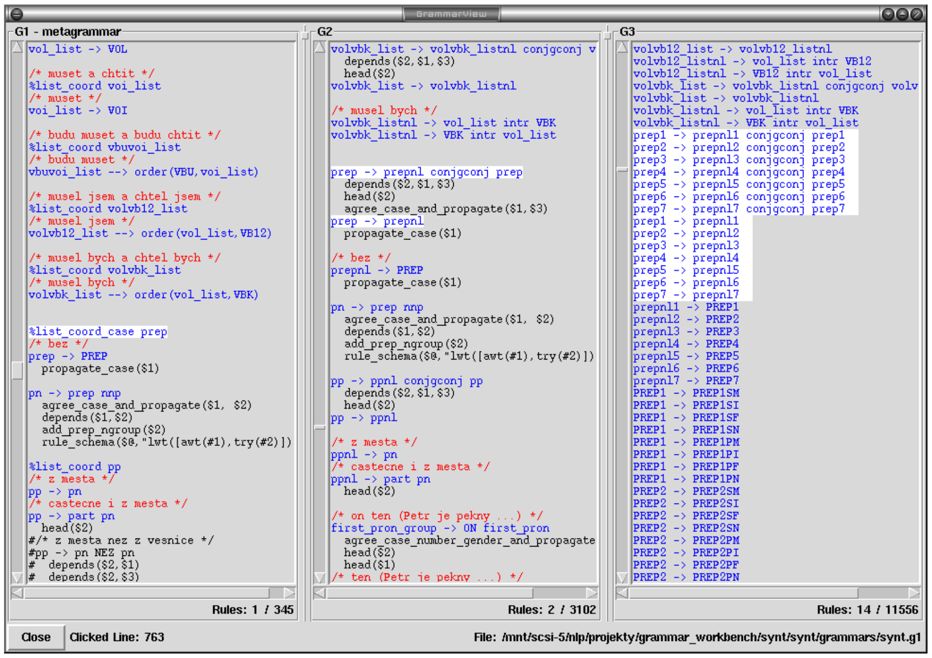
Fig. 2. An example of a VerbaLex verb frame

The Second Grammar Form (G2) Several pre-defined grammatical tests and procedures are used in the description of contextual actions associated with each grammatical rule of the system. The pruning actions include:

– grammatical case test for particular words and noun groups
– agreement test of case in prepositional construction
– agreement test of number and gender for relative pronouns
– agreement test of case, number and gender for noun groups
– type checking of logical constructions

Expanded Grammar Form (G3) The feature agreement tests can be transformed into context-free rules. For instance in Czech, similar to other Slavic languages, we have 7 grammatical cases (nominative, genitive, dative, accusative, vocative, locative and instrumental), two numbers (singular and plural) and three genders (masculine, feminine and neuter), in which masculine exists in two forms — animate and inanimate. Thus, e.g., we get 56 possible variants for a full agreement between two constituents.

The Figure 3 illustrates generative construct `%list_coord_case_prep` in G1, that produces two context-free rules with pruning actions in G2 and fourteen context-free rules in G3. The grammars are displayed by the GrammarView module, which is part of the Grammar Development Workbench environment, see the Section 6.



G1 - metagrammar

```
vol_list -> VOL

/* muset a chtit */
%list_coord voi_list
/* muset */
voi_list -> VOI

/* budu muset a budu chtit */
%list_coord vbuvoi_list
/* budu muset */
vbuvoi_list --> order(VBU,voi_list)

/* musel jsem a chtel jsem */
%list_coord volvb12_list
/* musel jsem */
volvb12_list --> order(vol_list,VB12)

/* musel bych a chtel bych */
%list_coord volvbk_list
/* musel bych */
volvbk_list --> order(vol_list,VBK)


%list_coord_case prep
/* bez */
prep -> PREP
  propagate_case($1)

pn -> prep nnp
  agree_case_and_propagate($1, $2)
  depends($1,$2)
  add_prep_ngroup($2)
  rule_schema($@,"lwt([awt(#1),try(#2)])

%list_coord pp
/* z mesta */
pp -> pn
/* castecne i z mesta */
pp -> part pn
  head($2)
#/* z mesta nez z vesnice */
#pp -> pn NEZ pn
#  depends($2,$1)
#  depends($2,$3)
```

Rules: 1 / 345

G2

```
volvbk_list -> volvbk_listnl conjgconj v
  depends($2,$1,$3)
  head($2)
volvbk_list -> volvbk_listnl

/* musel bych */
volvbk_listnl -> vol_list intr VBK
volvbk_listnl -> VBK intr vol_list


prep -> prepnl conjgconj prep
  depends($2,$1,$3)
  head($2)
  agree_case_and_propagate($1,$3)
prep -> prepnl
  propagate_case($1)

/* bez */
prepnl -> PREP
  propagate_case($1)

pn -> prep nnp
  agree_case_and_propagate($1, $2)
  depends($1,$2)
  add_prep_ngroup($2)
  rule_schema($@,"lwt([awt(#1),try(#2)])

pp -> ppnl conjgconj pp
  depends($2,$1,$3)
  head($2)
pp -> ppnl

/* z mesta */
ppnl -> pn
/* castecne i z mesta */
ppnl -> part pn
  head($2)

/* on ten (Petr je pekny ...) */
first_pron_group -> ON first_pron
  agree_case_number_gender_and_propagate
  head($2)
  head($1)
/* ten (Petr je pekny ...) */
```

Rules: 2 / 3102

G3

```
volvb12_list -> volvb12_listnl
volvb12_listnl -> vol_list intr VB12
volvb12_listnl -> VB12 intr vol_list
volvbk_list -> volvbk_listnl conjgconj volv
volvbk_list -> volvbk_listnl
volvbk_listnl -> vol_list intr VBK
volvbk_listnl -> VBK intr vol_list
prep1 -> prepnl1 conjgconj prep1
prep2 -> prepnl2 conjgconj prep2
prep3 -> prepnl3 conjgconj prep3
prep4 -> prepnl4 conjgconj prep4
prep5 -> prepnl5 conjgconj prep5
prep6 -> prepnl6 conjgconj prep6
prep7 -> prepnl7 conjgconj prep7
prep1 -> prepnl1
prep2 -> prepnl2
prep3 -> prepnl3
prep4 -> prepnl4
prep5 -> prepnl5
prep6 -> prepnl6
prep7 -> prepnl7
prepnl1 -> PREP1
prepnl2 -> PREP2
prepnl3 -> PREP3
prepnl4 -> PREP4
prepnl5 -> PREP5
prepnl6 -> PREP6
prepnl7 -> PREP7
PREP1 -> PREP1SM
PREP1 -> PREP1SI
PREP1 -> PREP1SF
PREP1 -> PREP1SN
PREP1 -> PREP1PM
PREP1 -> PREP1PI
PREP1 -> PREP1PF
PREP1 -> PREP1PN
PREP2 -> PREP2SM
PREP2 -> PREP2SI
PREP2 -> PREP2SF
PREP2 -> PREP2SN
PREP2 -> PREP2PM
PREP2 -> PREP2PI
PREP2 -> PREP2PF
PREP2 -> PREP2PN
```

Rules: 14 / 11556

Clicked Line: 763

File: /mnt/scsi-5/nlp/projekty/grammar_workbench/synt/synt/grammars/synt.g1

Fig. 3. Generative construct `%list_coord_case_prep` in the grammar G1 and the appropriate (generated) rules and actions in G2 and G3.

The number of rules naturally grows in the direction G1 $<$ G2 $<$ G3. The current numbers of rules in the three grammar forms are 253 in G1, 3091 in G2 and 11530 in G3, but the grammar is still being developed and enhanced.

In the current stage of the meta-grammar development, we have achieved an average of 92.08 % coverage[1] with 83.7 % cases where the correct syntactic tree was present in the result. However, the process of determining the correct tree is still premature.

5 Parser

The parsing process consists of several stages. Firstly, the packed shared forest [19] is produced by the standard CF parsing algorithm. Then the contextual constraints are applied. Finally, the trees are ordered by an efficient tree rank computation counted over the chart structure and n trees with highest ranks are selected. The order of the last two steps (constraints application and tree rank computation) can be reversed. All functions are implemented as modules that can be modified as needed or even substituted with different implementation. For example, three different parsing algorithms have been implemented. All algorithms use identical internal data structures (Earley's top-down and

[1] Measured on 10.000 Czech corpus sentences with an average time of 276 mili-seconds per sentence.

bottom-up chart parser [19], our variant of head-driven chart parser [12] and Tomita's GLR [20]). These implementations produce the same structures, thus applying contextual constraints or selecting n best trees can be shared among them.

It was shown [21] that parsing is in general case NP-complete problem if grammars are allowed to have agreement features. The pruning constraints in `synt` are weaker than general feature structures. It allows an efficient implementation with the following properties. A node in the derivation tree has only limited number of values (e.g. the cardinality of the set for noun groups in our system is max. $56 = 7\text{cases} \times 2\text{nouns} \times (3+1)\text{genders}$). In `synt` instead of usual pruning of the original packed share forest, a new forest of values is built during the analysis. The worst case time complexity for one node in the forest of values is therefore 56^{δ}, where δ is the length of the longest right-hand side of grammar rules. Note that this complexity is independent of the input sentence.

6 Grammar Development Workbench

All the above described tools and systems are presented to a user by means of a front-end tool – the *Grammar Development Workbench (GDW)*. Because most of the platform components (`synt`, `ajka`, ...) are controlled only from command-line, the GDW graphical user interface has been created to allow users comfortable and well arranged work with the tools.

GDW has been created by FI MUNI student Radek Vykydal [13] under supervision of Aleš Horák. The system is still under development and it is now extensively tested by linguists. GDW consists of five modules:

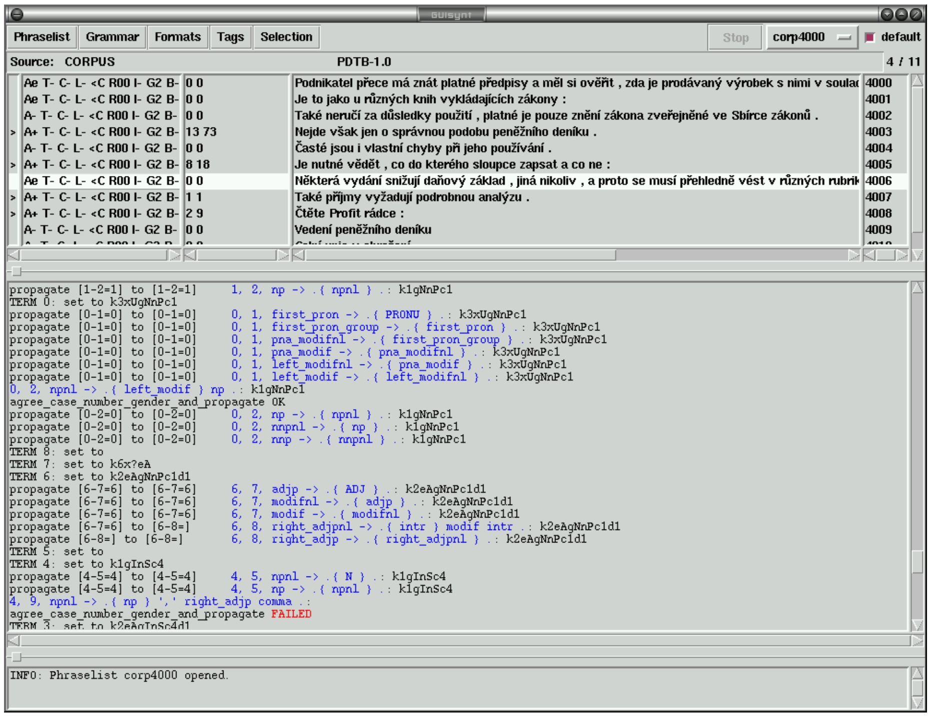
- Gsynt – graphical user interface of the `synt` parser.
- TreeView – viewer for resulting syntactic trees.
- ChartView – browser of resulting chart structure.
- GrammarView – grammar forms viewer.
- TBAdmin – tree bank creation tool.

The linguistic work concentrates on the methodology of grammar development with respect to real-world natural language texts. The most important GDW tasks are enhancing the meta-grammar and creation of a tree bank.

For building a tree bank of correct syntactic trees, the following steps are conducted. First of all, a corpus sentence is analysed by the Gsynt module. Figure 4 shows basic window with sentences from a corpus (PDTB-1.0 [16] in this case). The selected sentence is analyzed by the `synt` parser and the output of the parser is displayed. Several additional information about sentences is presented in the window.

The syntactic trees are displayed by the TreeView module. The trees are all consistent with the imposed meta-grammar and they are ordered by the computed tree ranks. However, the ordering is still premature and the correct tree is often not in the first positions yet. The number of syntactic trees is reduced by successive specification of pruning constraints. The resulted syntactic tree is added to the tree bank at the end.

Enhancing the meta-grammar lies in determining new rules for an unaccepted sentence. By means of the ChartView module, the problematic sentence construct that is



```
Source:  CORPUS                         PDTB-1.0                                                                  4 / 11
  Ae T- C- L- <C R00 I- G2 B-  0 0    Podnikatel přece má znát platné předpisy a měl si ověřit , zda je prodávaný výrobek s nimi v soulad  4000
  Ae T- C- L- <C R00 I- G2 B-  0 0    Je to jako u různých knih vykládajících zákony :                                                     4001
  A- T- C- L- <C R00 I- G2 B-  0 0    Také neručí za důsledky použití , platné je pouze znění zákona zveřejněné ve Sbírce zákonů .        4002
> A+ T- C- L- <C R00 I- G2 B-  13 73  Nejde však jen o správnou podobu peněžního deníku .                                                  4003
  A- T- C- L- <C R00 I- G2 B-  0 0    Časté jsou i vlastní chyby při jeho používání .                                                      4004
> A+ T- C- L- <C R00 I- G2 B-  8 18   Je nutné vědět , co do kterého sloupce zapsat a co ne :                                              4005
  Ae T- C- L- <C R00 I- G2 B-  0 0    Některá vydání snižují daňový základ , jiná nikoliv , a proto se musí přehledně vést v různých rubrik 4006
> A+ T- C- L- <C R00 I- G2 B-  1 1    Také příjmy vyžadují podrobnou analýzu .                                                             4007
> A+ T- C- L- <C R00 I- G2 B-  2 9    Čtěte Profit rádce :                                                                                 4008
  A- T- C- L- <C R00 I- G2 B-  0 0    Vedení peněžního deníku                                                                              4009
```

```
propagate [1-2=1] to [1-2=1]     1, 2, np -> .{ npnl } .: k1gNnPc1
TERM 0: set to k3xUgNnPc1
propagate [0-1=0] to [0-1=0]     0, 1, first_pron -> .{ PRONU } .: k3xUgNnPc1
propagate [0-1=0] to [0-1=0]     0, 1, first_pron_group -> .{ first_pron } .: k3xUgNnPc1
propagate [0-1=0] to [0-1=0]     0, 1, pna_modifnl -> .{ first_pron_group } .: k3xUgNnPc1
propagate [0-1=0] to [0-1=0]     0, 1, pna_modif -> .{ pna_modifnl } .: k3xUgNnPc1
propagate [0-1=0] to [0-1=0]     0, 1, left_modifnl -> .{ pna_modif } .: k3xUgNnPc1
propagate [0-1=0] to [0-1=0]     0, 1, left_modif -> .{ left_modifnl } .: k3xUgNnPc1
0, 2, npnl -> .{ left_modif } np .: k1gNnPc1
agree_case_number_gender_and_propagate OK
propagate [0-2=0] to [0-2=0]     0, 2, np -> .{ npnl } .: k1gNnPc1
propagate [0-2=0] to [0-2=0]     0, 2, nnpnl -> .{ np } .: k1gNnPc1
propagate [0-2=0] to [0-2=0]     0, 2, nnp -> .{ nnpnl } .: k1gNnPc1
TERM 8: set to
TERM 7: set to k6x?eA
TERM 6: set to k2eAgNnPc1d1
propagate [6-7=6] to [6-7=6]     6, 7, adjp -> .{ ADJ } .: k2eAgNnPc1d1
propagate [6-7=6] to [6-7=6]     6, 7, modifnl -> .{ adjp } .: k2eAgNnPc1d1
propagate [6-7=6] to [6-7=6]     6, 7, modif -> .{ modifnl } .: k2eAgNnPc1d1
propagate [6-7=6] to [6-8=]      6, 8, right_adjpnl -> .{ intr } modif intr .: k2eAgNnPc1d1
propagate [6-8=] to [6-8=]       6, 8, right_adjp -> .{ right_adjpnl } .: k2eAgNnPc1d1
TERM 5: set to
TERM 4: set to k1gInSc4
propagate [4-5=4] to [4-5=4]     4, 5, npnl -> .{ N } .: k1gInSc4
propagate [4-5=4] to [4-5=4]     4, 5, np -> .{ npnl } .: k1gInSc4
4, 9, npnl -> .{ np } ',' right_adjp comma .:
agree_case_number_gender_and_propagate FAILED
TERM 3: set to k2eAgInSc4d1
```

```
INFO: Phraselist corp4000 opened.
```

Fig. 4. The Gsynt module window showing an analysis of a sentence from corpus.

not covered by the meta-grammar can be located. Then the exact specification of the needed rule(s) for the uncovered construct is determined and entered in the Grammar-View module, see the Figure 3.

For more information about the GDW project see the project documentation page http://nlp.fi.muni.cz/projects/grammar_workbench/manual-en/.

7 Conclusion

This paper has described several tools, methods and systems that together form a platform for full syntactic analysis of free-word order natural language, the Czech language. The development of the platform components is definitely not yet completed, however, already at this stage the system offers very high coverage on common corpus sentences while keeping the analysis time on the level suitable for batch processing of large texts.

Acknowledgments

This work has been partly supported by Czech Science Foundation under the project 201/05/2781 and by the Ministry of Education of the Czech Republic in the National Research Programme II project 2C06009.

References

1. Charniak, E.: Immediate-head parsing for language models. In: Proceedings of the 39th Annual Meeting of the Association of Computational Linguistics, Toulouse, France (2001)
2. Collins, M.: Head-driven Statistical Models for Natural Language Parsing. PhD thesis, Computer Science Department, University of Pennsylvenia, Philadelphia (1999)
3. R Debusmann, . In: A Parser System for Extensible Dependency Grammar, Loria, Nancy France (2003) 103–106
4. Kruijff, G.: 3-phase grammar learning. In: Proceedings of Workshop on Ideas and Strategies for Multilingual Grammar Development, Vienna Austria (2003)
5. Horák, A., Kadlec, V.: New meta-grammar constructs in czech language parser synt. In: TSD. (2005) 85–92
6. Maxwell III, J.T., Kaplan, R.M.: The interface between phrasal and functional constraints. In Rosner, M., Rupp, C.J., Johnson, R., eds.: Proceedings of the Workshop on Constraint Propagation, Linguistic Description, and Computation, Instituto Dalle Molle IDSIA, Lugano (1991) 105–120 Also in Computational Linguistics, Vol. 19, No. 4, 571–590, 1994.
7. Sedláček, R.: Morphemic Analyser for Czech. PhD thesis, Masaryk University, Brno, Czech Republic (2005)
8. Hlaváčková, D., Horák, A.: Verbalex – new comprehensive lexicon of verb valencies for czech. In: Proceedings of the Slovko Conference, Bratislava, Slovakia (2005)
9. Horák, A., Smrž, P.: Best analysis selection in inflectional languages. In: Proceedings of the 19th international conference on Computational linguistics, Taipei, Taiwan, Association for Computational Linguistics (2002) 363–368
10. Kadlec, V., Smrž, P.: PACE - parser comparison and evaluation. In: Proceedings of the 8th International Workshop on Parsing Technologies, IWPT 2003, Le Chesnay Cedex, France, INRIA, Domaine de Voluceau, Rocquencourt (2003) 211–212
11. Horák, A., Kadlec, V.: New meta-grammar constructs in czech language parser synt. In: Proceedings of Text, Speech and Dialogue 2005, Karlovy Vary, Czech Republic, Springer-Verlag (2005) 85–92
12. Kadlec, V., Smrž, P.: How many dots are really needed for head-driven chart parsing? In: Proceedings of SOFSEM 2006, Czech Republic, Springer-Verlag (2006) 483–492
13. Vykydal, R.: Nástroje pro vývoj gramatik přirozeného jazyka (tools for developing natural language grammars). Master's thesis, Faculty of Informatics, Masaryk University, Brno, Czech Republic (2005) in Czech.
14. Hlaváčková, D., Horák, A., Kadlec, V.: Exploitation of the verbalex verb valency lexicon in the syntactic analysis of czech. In: Proceedings of Text, Speech and Dialogue 2006, Brno, Czech Republic, Springer-Verlag (2006) 79–85
15. Žabokrtský, Z., Lopatková, M.: Valency Frames of Czech Verbs in VALLEX 1.0. In Meyers, A., ed.: HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation. (2004) 70–77
16. Hajič, J.: Building a syntactically annotated corpus: The Prague Dependency Treebank. In: Issues of Valency and Meaning, Prague, Karolinum (1998) 106–132
17. Pala, K., Sevecek, P.: Valence českých sloves (Valencies of Czech Verbs). In: Proceedings of Works of Philosophical Faculty at the University of Brno, Brno, Masaryk University (1997) 41–54
18. Dang, H.T., Kipper, K., Palmer, M., Rosenzweig, J.: Investigating regular sense extensions based on intersective levin classes. In: Proceedings of Coling-ACL98, Montreal CA (August 11-17, 1998) http://www.cis.upenn.edu/∼mpalmer/.
19. Earley, J.: An efficient context-free parsing algorithm. In: Communications of the ACM. Volume 13. (1970) 94–102
20. Tomita, M.: Efficient Parsing for Natural Languages: A Fast Algorithm for Practical Systems. Kluwer Academic Publishers, Boston, MA (1986)
21. Barton, G.E., Berwick, R.C., Ristad, E.S.: Computational complexity and natural language. MIT Press, Cambridge, Massachusetts (1987)