Parsing System with Contextual Constraints

Vladimír Kadlec

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

Abstract. The aim of the presented work is to design algorithms and methods for an effective and robust syntactic analysis of natural languages. The algorithms are language-independent, any language with an appropriate grammar can be modeled. The analysis of sentences by the described system is based on context-free grammar for a given language supplemented by context sensitive structures. The internal representation of derivation trees allows to apply contextual constraints, e.g. case agreement fulfillment. The evaluation of semantic actions and contextual constraints helps us to reduce a huge number of derivation trees and we are also able to calculate some new information, which is not contained in the context-free part of the grammar. Also *n*-best trees (according to a tree rank, e.g probability) can be selected. This is an important feature for linguistics developing a grammar by hand.

1 Introduction

Syntactic analysis is a "corner-stone" of applications for automated processing of texts in natural languages. Any machine translation application, an automatic grammar checker or information retrieval system must be capable of understanding the structure of a sentence. Recognition of the sentence structure is called *parsing*.

The analysis of sentences by the described system is based on context-free grammar for the given language. Context-free parsing techniques are well suited to be incorporated into real-world nature language processing systems because of their time efficiency and low memory requirements. Though, it is known that some natural language phenomena cannot be handled with the context-free grammar formalism, researchers often use the context-free backbone as the core of their grammar formalism and enhance it with context sensitive feature structures (e.g. [1]).

2 System Overview

Described system consists of several independent modules. The modular design makes the system easily extensible and rather flexible. Figure 1 shows the data flow through the system.

There are several inputs to the system:

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2007, pp. 13–20, 2007. © Masaryk University, Brno 2007

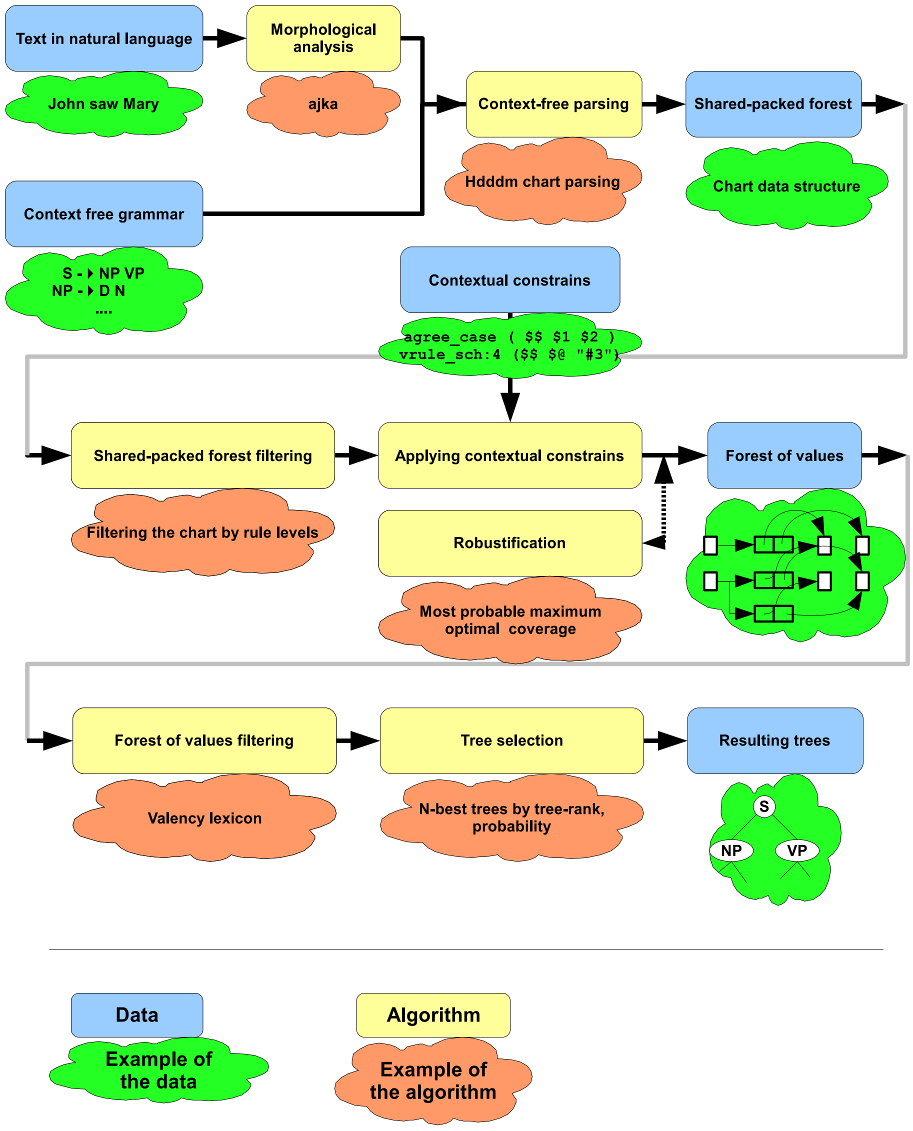

Fig. 1. Parsing System with Contextual Constraints

– A sentence in a natural language.
– Context-free (CF) grammar.
– Semantic actions and contextual constraints for grammar rules.

Words in the input sentence can be optionally tagged. If they are not tagged, then the internal "tagger" is used. The notation of "tagger" is not correct here, because we leave ambiguities in tags. For the Czech language, morphological analyzer `ajka` [2] is used to create tags. For other languages the tags are usually read from an static lexicon. These tags are stored as "values" (see below) for every word. The terminals (or sometimes called pre-terminals) for given context-free grammar are created by simplification the tags, e.g. using only word category as a terminal.

Once the terminals are created the context-free parsing algorithm is run. This algorithm produces all possible derivation trees at the output with respect to the input sentence and input grammar. All these derivation trees are stored in a data structure based on shared-packed forest [3]. Because a chart parser is used in our system [4], the derivation trees are stored as a chart data structure [5,6] directly. Any context-free parsing algorithm could be used, the modularity of the system allows us compare the effectiveness of these algorithms easily [7].

All derivation trees created in the previous step can be filtered by some basic filter, that cuts some trees off. In this step only basic filtering "compatible" with the shared-packed forest data structure is allowed. E.g. only a whole sub-forests can be removed. The example of such filtration is in Section 3.

The next step is application of contextual constraints and semantic actions. In this step a new data structure is created, a "forest of values". The forest of values is created by a bottom-up recursive run of semantic actions, see Section 4.

If the input sentence cannot be generated by the input grammar, i.e. there is no derivation tree at the output of the context-free parsing algorithm, the system offers a robust module. In this case, the contextual constraints and semantic actions are applied on every derivation sub-tree in the shared-packed forest. Then the robust algorithm presented in [8] is used to get the derivation tree(s).

The resulting forest of values can be further filtered by constraints, that work with the whole forest, not only with local values. The example of this global filtering is usage lexicon of verb valencies VerbaLex, see [9].

In the end, the derivation trees are generated from the filtered forest of values. Only one or several "best" derivation trees can be created, with respect to the ranking function, e.g. a probability of the tree could be used as one input to the ranking function.

In the following sections, the above ideas are described in more detail.

3 Shared-Packed Forest filtering

This filtering is used only to remove some data from the structure, no new information is added. This technique is a step preceding the filtering by contextual constraints, see Figure 1 in Section 2 for an overview of the whole system. The resulting shared-packed forest is always sub-forest of the original. That means that only simple transformations such as removing a node is done here.

3.1 Filtering by rule levels

One of possible filtering the shared-packed forest is a local (with respect to the node of the forest) filtering based on what we call "rule levels". The rule level is a function, that assigns a natural number to every grammar rule. In the following the term "a rule level of a grammar rule" denotes the resulting number of application this function to the rule.

The idea is, that for some grammar rules: if the specific grammar rule succeeds during the parse process, then an application of some other rule (with the same non-terminal on the right hand side of the rule) is wrong. To be more precise, if the specific grammar rule covers the same input as some other grammar rule beginning with the same non-terminal, then the rule with lower rule level is refused.

The chart structure [5] represents the shared-packed forest. So the filtering method is described in terms of the chart parsing: If there are edges $[A \rightarrow \bullet\alpha\bullet, i, j]$ and $[A \rightarrow \bullet\beta\bullet, i, j]$ in the chart, then delete the edge with the grammar rule with the lower rule level. If the edges have the same rule level, keep them both in the chart. Figure 2 shows an example of such rules, $w_1, w_2, \ldots, w_6$ represent the input sentence.

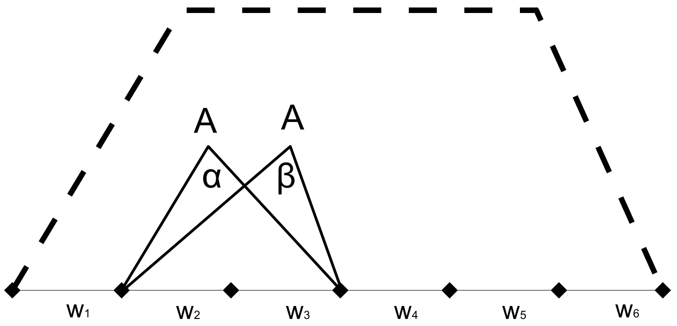

Fig. 2. Filtering by rule levels. Two sub-forests with grammar rules $A \rightarrow \alpha$ and $A \rightarrow \beta$ in their roots. One of them is filtered out, if these rules has a different rule level set.

Notice that this kind of filtering is different from a probability of the grammar rule. The presented method is local to the specific node in the shared-packed forest. By default all grammar rules have the same rule level. The rule levels are set by hand and only in very specific cases. Actually, only one rule in our grammar for Czech [10] has non-default rule level. Only small number of experiments were performed, because this method is new to our system.

4 Contextual Constraints and Semantic Actions

Our main problem with the context-free (CF) parsing is, that there are too many derivation trees for a given input sentence. The contextual constraints are mainly used to prune incorrect derivation trees from the CF parsing result. Also some additional information can be computed by these constraints, that is why we also call them "semantic actions". In the following the term "contextual constraint" has the same meaning as the term "semantic action". Our algorithms for CF parsing generates the chart structure, thus we use the word "chart" to denote "a result of the CF parsing".

See Figure 1 to have a better view, in which part of the parsing system the constraints are computed.

The contextual constraints (or actions) defined in the grammar can be divided into four groups:

1. rule-tied actions
2. case agreement constraints
3. post-processing actions
4. actions based on derivation tree

The example of a rule-tied action is a rule-based probability estimation. Case agreement constraints serve as chart pruning actions. The case agreement constraints represent the functional constraints, whose processing can be interleaved with that of phrasal constraints.

The post-processing actions are not triggered until the chart is already completed. Actions on this level are used mainly for computation of analysis probabilities for a particular input sentence and particular analysis. Some such computations (e.g. verb valency probability) demand exponential resources for computation over the whole chart structure. This problem is solved by splitting the calculation process into the pruning part (run on the level of post-processing actions) and the reordering part, that is postponed until the actions based on derivation tree.

The actions that do not need to work with the whole chart structure are run after the best or *n* most probable derivation trees have been selected. These actions are used, for example, for determination of possible verb valencies within the input sentence, which can produce a new ordering of the selected trees, or for the logical analysis of the sentence [11].

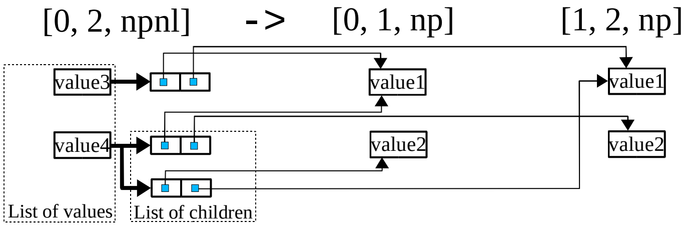

Fig. 3. Example of the forest of values.

4.1 Representation of values

It was shown that parsing is in general NP-complete if grammars are allowed to have agreement features [12]. But the pruning constraints in our system are weaker than for example general feature structures [13].

We allow a a node in the derivation tree to have only limited number of features. We call the features "values", because they rise as results of our semantic actions. E.g. the number of values for noun groups in our system is at most 56. To compute the values, we build a new structure, a forest of values, instead of pruning or extending the original chart.

The forest of values is computed by the depth-first walk through the chart structure. The chart can be viewed as oriented graph. Every edge in the chart is passed only once, the edge can generate at most one node in the new forest of values.

The value is computed as a result of the semantic action – for the grammar rule given by the current edge. The parameters for the semantic action are filled from the values on lower level, "lower" with respect to the derivation tree, i.e. closer to the leaves of the tree. So also arguments of the semantic action are limited by the limit (e.g. 56 possibilities in our case). Because there could be more than one derivation tree containing the current edge, all possible combination of values are passed to the semantic action. 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 grammar rule. Notice that this complexity is independent of the number of words in input sentence.

The values in the forest of values are linked with the edges backwards. An edge contains a single linked list of its values. Each value holds a single linked list of its children. The child is one dimensional array of values. This array represents one combination of values that leads to the parent value. Notice that there can be more combinations of values that lead to the same value. The i-th cell of the array contains a reference to a value from i-th symbol on the RHS of the corresponding grammar rule. The i-th symbol has not to be used to compute the parent value. We use only reference to the edge from such unused cell.

The Figure 3 shows an example representing the rule $npnl \rightarrow np\ np$ and containing three edges ($[0, 2, npnl \rightarrow \bullet np\ np \bullet]$, $[0, 1, np \rightarrow \bullet\alpha\bullet]$, $[1, 2, np \rightarrow \bullet\beta\bullet]$). The right hand sides of each rule are not shown in the figure, they play no role here. $np \rightarrow \alpha$ and $np \rightarrow \beta$ are some rules in the input grammar.

Each np edge contains two values, *value1* and *value2*. This gives us four possible combinations. The semantic action computes from combinations *value1* × *value2* and *value2* × *value1* the same value *value4*. The combination *value2* × *value2* was classified as incorrect (by the action – contextual constraint), so it is not here.

4.2 Generation of a grammar with values

It is possible to create CF grammar, without our contextual constraints, which generates the same derivation trees as the CF grammar supplemented by the constraints. In the following, a method, that for the given input generates a such CF grammar without values, is provided. This allows us to compare our system, that is able to evaluate the constraints, with other systems able to work only with "pure" CF grammars.

We use the following procedure for every inactive edge $[i, j, A \rightarrow X_1 X_2 ... X_n \bullet]$ in the chart:

- for every value v in the edge, we generate the rule: $A \rightarrow A\_value$, where $value$ is an unique textual representation of the value v.
- for every child of the value v, we generate the rule: $A\_value \rightarrow X_1^{'} X_2^{'} ... X_n^{'}$, where $X\_i^{'}$ is:
  - $X_i\_value_i$ if a value $value_i$ from i-th non-terminal is used to compute the value v.
  - X_i otherwise.

Duplicate rules are removed.

Why are the actions and semantic constraints used when they can be replaced by a grammar with values? There are three main reasons. First of all, the grammar with values for all possible inputs would be extremely large, even if the domain range is limited, e.g. by 56 in our case. Secondly, the actions can be easily changed and debugged when computed separately. The third reason is that some of our experiments use semantic actions with unlimited domain range and these actions cannot be substituted by the grammar.

5 Conclusions

In this work, the language independent parsing system is presented. It is based on context-free parser supplemented by contextual constraints and semantic actions.

The evaluation of semantic actions and contextual constraints helps us to reduce a huge number of derivation trees and we are also able to calculate some new information which is not covered in the context-free part of the grammar. The dependency graph or filtering by valency lexicon are examples of such information. The experiments with dependency graphs are at the beginning. But even for some kinds of short non-projective sentences, the correct dependencies can be generated within our approach as well.

All described algorithms are integrated in the parsing system synt [10,14]. Future research is aimed at the experiments with verb valences and lexicon of verb valencies for the Czech VerbaLex.

Acknowledgements This work has been partly supported by the Academy of Sciences of Czech Republic under the project 1ET100300414 and by the Ministry of Education of CR within the Center of basic research LC536 and by the Czech Science Foundation under the project 201/05/2781.

References

1. Neidle, C.: Lexical-Functional Grammar (LFG). In Asher, R.E., ed.: Encyclopedia of Language and Linguistics. Volume 3. Pergamon Press, Oxford (1994) 2147–2153.
2. Sedláček, R.: Morphemic Analyser for Czech. Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno, Czech Republic (2005).
3. Tomita, M.: Efficient Parsing for Natural Languages: A Fast Algorithm for Practical Systems. Kluwer Academic Publishers, Boston, MA (1986).
4. 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.
5. Kay, M.: Algorithm schemata and data structures in syntactic processing. In: Report CSL-80-12, Palo Alto, California, Xerox PARC (1989).
6. Earley, J.: An efficient context-free parsing algorithm. In: Communications of the ACM. Volume 13. (1970) 94–102.
7. 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.
8. Kadlec, V., Ailomaa, M., Chappelier, J.C., Rajman, M.: Robust stochastic parsing using optimal maximum coverage. In: Proceedings of The International Conference Recent Advances In Natural Language Processing (RANLP) 2005, Shoumen, Bulgaria, INCOMA (2005) 258–263.
9. 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) 85–92.
10. 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.
11. Horák, A.: Analysis of Knowledge in Sentences. Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno, Czech Republic (2002).
12. Barton, G.E., Berwick, R.C., Ristad, E.S.: Computational complexity and natural language. MIT Press, Cambridge, Massachusetts (1987).
13. Kay, M.: Parsing in functional unification grammar. In: Natural Language Parsing, England, Cambridge (1985) 251–278.
14. Horák, A., Kadlec, V.: Platform for Full-Syntax Grammar Development Using Meta-grammar Constructs. In: Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation, Beijing, China, Tsinghua University Press (2006) 311–318.