Incremental Parser for Czech

Pavel Smrž and Vladimír Kadlec
Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno, Czech Republic
E-mail: {smrz,xkadlec}@fi.muni.cz

Abstract

The presented paper deals with an efficient implementation of incremental bottom-up chart parser for Czech. We shortly introduce the system based on a metagrammar that generates a context-free backbone supplemented by contextual constraints. The main part of the paper focuses on the way the pruning constraints are evaluated in the incremental parsing mode. The algorithm takes advantage of a special form of feature structures employed in our grammar, namely the limited number of values that can be produced for each type of the constraints. Instead of pruning the original packed share forest, the parser builds a new forest of values resulting from the evaluation. The described method enables efficient parsing even for extremely ambiguous grammars typical for free word order languages.

1 Introduction

The efficient implementation of robust syntactic analyzer for free Czech text was described in [SH99, SH00, HKS02]. As Czech is a typical representative of the Slavic language family characterized by free word order, grammars aiming at covering a reasonable portion of the most frequent Czech language phenomena tend to be extremely ambiguous (e. g. 10^{27} possible analyses). Various parsing algorithms — CYK [Kas65, You67, AU72], GLR [Tom86], Earley [Ear70], left-corner chart parser [Kay89a, Mea83], head-driven chart parser [Kay89b, SS89]) have been therefore applied with the goal to find an optimum for our needs.

This paper presents an incremental variant of the Czech parser that is based on bottom-up head-driven chart parsing algorithm. The next section briefly summarizes the core functionality of the special kind of head-driven dependent dot move algorithm integrated into our system. The third section discusses the form of feature structures [Bre85, GKPS85] in our grammar and the algorithm that implements their efficient computation.

2 Parsing algorithm

The parsing technique employed in our experiments is based on the head driven approach similar to [SS89, SodA93] with improvements in the process of confirmation of viable hypotheses. The

HDddm (head-driven with dependent dot move) parsing technique refers to the fact that the move of one “dot” in the head-driven parsing step is dependent on the opposite move of the other one.

The head of a grammar rule is a symbol from the right hand side. For example, the second nonterminal (`np`) is denoted as the head symbol in the following grammar rule

```
np -> left_modif np
```

The epsilon rule has a special head symbol ϵ. The edge in the head driven parser is a triplet [$A \rightarrow \alpha_\bullet\beta_\bullet\gamma$, i, j], where i, j are integers, $0 \leq i \leq j \leq n$ for currently n words analyzed from the input sentence and $A \rightarrow \alpha\beta\gamma$ is a rule in the input grammar. The direction of the parsing process does not move unidirectionally from left to right, but it starts at the head of the grammar rule.

The parsing algorithm can be summarized by the following schema, where the symbol G stands for the input grammar with a set of rules P and the root symbol S. $a_1, a_2, ..., a_i, ...$ are input words (preterminals).

Initialization phase (processed once at the very beginning)

1. for each $A \rightarrow \epsilon \in P$ add edge $[A \rightarrow {}_{\bullet\bullet}, 0, 0]$ to the chart.

Initialization phase for the i-th input word

1. for each $A \rightarrow \epsilon \in P$ add edge $[A \rightarrow {}_{\bullet\bullet}, i, i]$ to the chart.
2. for each $A \rightarrow \alpha\underline{a_i}\beta \in P$ (a_i is the head of the rule) add edge $[A \rightarrow \alpha_\bullet a_{i\bullet}\beta,\ i-1,\ i]$ to the chart.

Iteration phase

1. if edge E in the chart is in the form [$A \rightarrow {}_\bullet\alpha_\bullet$, j, k], then for each edge:

   [$B \rightarrow \beta_\bullet\gamma_\bullet A\delta$, i, j] in the chart, create edge [$B \rightarrow \beta_\bullet\gamma A_\bullet\delta$, i, k].

   [$B \rightarrow \beta A_\bullet\gamma_\bullet$, k, l] in the chart, create edge [$B \rightarrow \beta_\bullet A\gamma_\bullet$, j, l].

2. if E is in the form [$B \rightarrow \beta_\bullet\gamma_\bullet A\delta$, i, j], then for each edge [$A \rightarrow {}_\bullet\alpha_\bullet$, j, k] in the chart, create edge [$B \rightarrow \beta_\bullet\gamma A_\bullet\delta$, i, k].
3. if E is in the form [$B \rightarrow \beta A_\bullet\gamma_\bullet$, k, l], then for each edge [$A \rightarrow {}_\bullet\alpha_\bullet$, j, k] in the chart, create edge [$B \rightarrow \beta_\bullet A\gamma_\bullet$, j, l].
4. if E is in the form [$A \rightarrow \beta_\bullet\gamma_\bullet a_{j+1}\delta$, i, j], then create edge [$A \rightarrow \beta_\bullet\gamma a_{j+1\bullet}\delta$, i, j+1].
5. if E is in the form [$A \rightarrow \beta a_{i\bullet}\gamma_\bullet$, i, j], then create edge [$A \rightarrow \beta_\bullet a_i\gamma_\bullet$, i-1, j].
6. if E is in the form [$A \rightarrow {}_\bullet\alpha_\bullet$, i, j], then for each rule $B \rightarrow \beta\ \underline{A}\ \gamma$ in the input grammar, create edge [$B \rightarrow \beta_\bullet A_\bullet\gamma$, i, j] (symbol A is the head of the rule).

N.B., that the left dot in the edge cannot move leftwards until the right dot moves to the right. The parser never creates edges like [$A \rightarrow \alpha_\bullet \beta \underline{A} \gamma_\bullet \delta$, *i, j*] for non empty β. This approach (similar to [SS89]) avoids the redundant analysis of such edges. The algorithm can handle with epsilon rules. This is important for us, because our CF grammar for Czech uses the epsilon rules[1]. On the other hand, the parser does not use any top-down filtering or "follow check" technique.

The efficiency of the parser depends to a considerable extent on the choice of grammar rule heads. The current positions of heads in our grammar have been chosen experimentally and they accords with the conception of the leading constituent in the traditional Czech grammars.

3 Incremental Evaluation of Contextual Constraints

3.1 Form of Rules

The incremental parser for Czech is based on a grammar comprising the context-free (CF) backbone and contextual constraints that can be specified for each CF rule. As the number of the rules is rather big (10,000 or 30,000 depending on the degree of grammar coverage), linguists do not work directly with this form. The set is generated from a specially designed metagrammar format [SH00] that enables to reduce the number of rules to a maintainable amount (approx. 300 metarules in the current system).

The generated grammar takes form of a lexicalized rule system in which (pre)terminals are present only in rules of the form $A \rightarrow a$. We adopt the terminology (and the format) of parser generators for programming languages and call the constraints semantic actions.

Every grammar rule has zero, one or more semantic actions. The actions are applied to:

- compute a value used by another action on a higher level;
- throw out incorrect analysis.

For example, the following grammar rule for genitive constructions in Czech has three semantic actions:

```
npnl -> np np +0.0784671532846715
  test_genitive ( $2 )
  propagate_all ( $$ $1 )
  depends:1 ( $$ $1 $2 )
```

First line contains a grammar rule with its probability computed from a treebank corpus. The semantic actions are specified on the next lines. Parameter $\$\$$ represents the return value, variable $\$n$ stores the value of the n-th nonterminal on the right hand side.

[1]The nonterminals of the form $A \Rightarrow^* \epsilon$ appear in our grammar very often, thus equivalent grammar without epsilon rules would be very huge.

3.2 Computation of the Semantic Actions

The parser used in our system is based on the chart structure so that we are able to compute the actions directly on the chart. The following text discusses the evaluation of the actions on chart edges.

The non-incremental variant of our parser takes advantage of an isolation of semantic constraint evaluation from the CF parsing phase. The execution of semantic actions is a separate step in the process of analysis — the actions are executed on the finished chart. This setting brings modularity and relative independence of the particular parsing algorithm. As the incremental parser is based on the strict bottom-up approach the system can interleave the processing of contextual constraints and the processing of the context-free backbone of the grammar.

It was shown in [BBR87] that, if grammars are allowed to have agreement features, parsing is NP-complete problem in general case. The pruning constraints in our system are weaker (see below) than general feature structures. Thus, we were able to design and implement an efficient method of their evaluation with the property that the number of values for each node in the derivation tree can be limited by a constant. For example, the cardinality of the set for noun groups in our system is at most $56 = 7 \times 2 \times (3 + 1)$ — seven 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.

Instead of pruning the finished chart (the chart coming as a result of the head driven parsing algorithm after i-th word) we build new *forest of values*. The actions are computed bottom-up (like e.g. in `bison` [CSH02]). Actions for lexical edges are executed first followed by the actions for higher level edges. The values are computed incrementally, i.e. we compute values only for new edges, that were created in the current (i-th) parsing step.

The worst-case time complexity for one node in the forest of values is C^δ, where C is the upper limit for the number of values in nodes (56 in the case discussed above) and δ is the length of the longest right-hand side of grammar rules. Note that the complexity is independent of the input sentence. The number of nodes is less than or equal to the number of inactive edges (nodes correspond to inactive edges, but some nodes can be pruned). Thus, the overall time complexity is $O(|G|C^\delta n^3)$.

Several experiments were performed that aimed at discovering an optimal strategy for evaluation of actions in the context of incremental parsing. The computation of actions for *every* inactive (complete) edge in the chart showed to be the best approach. We will explain why it is the case. Let us imagine the following situation: A successful parse for the current input sequence has been obtained. If there will be no additional word (i. e. no other input is available) then the evaluation of inactive edges, which are *not* used in successful parse(s), will be useless. However, our research shows that this situation is rather rare. The next input usually adds a word to the current state of the analysis and the parsing process benefits from the pre-computation of values for the actually unused edges.

The advanced mechanism of the parser allows to evaluate the actions even for currently active edges. A special structure is employed to identify the state when the parts that have not been analyzed yet will not be needed for the evaluation of an action. The following example demonstrates such a situation:

```
  A ->  B  C
action ( $$ $1 )
```

The rule $A \to BC$ has only one action `action`. There is edge $[0, 2, A \to {}_\bullet B_\bullet C]$ in the finished chart, but there is no edge $[2, k, C \to {}_\bullet \alpha_\bullet]$ and so there is no edge $[0, k, A \to {}_\bullet BC_\bullet]$. However the value for edge $[0, 2, A \to {}_\bullet B_\bullet C]$ can be computed, because `action` does not use the value from nonterminal C.

4 Conclusions and Future Directions

The presented robust parser for Czech provides a very good base for an integration into various NLP applications where the incrementality is a strict requirement. We currently work on a system that will combine the parser with a sophisticated language model for Czech (making account of the Czech rich morphology and agreement constraints). It should result in a mobile application for predictive writing of messages on cellular phones.

Future research will focus on efficiency issues concerning the output representation of the analysis. The current state of the parsing process is given by the set of maximal trees (trees that are not subtrees of another tree). The question is how to efficiently find all the trees. The inactive chart edges that are not referenced from the upper level are exactly the roots of maximal trees. Thus, all the maximal trees can be found in $O(k)$, where k is the number of edges. However, the algorithm could also label the edges during parsing to reduce the complexity of the process.

Acknowledgements

This work was supported by the Grant Agency of the Czech Republic Grant GACR 405/03/0913 and by the Ministry of Education of the Czech Republic Research Intent CEZ:J07/98:143300003.

References

[AU72] A.V. Aho and J.D. Ullman. *The Theory of Parsing, Translation and Compiling*, volume I: Parsing. Prentice-Hall, Englewood Cliffs, N.J., 1972.

[BBR87] G. E. Barton, R. C. Berwick, and E. S. Ristad. *Computational complexity and natural language*. MIT Press, Cambridge, Massachusetts, 1987.

[Bre85] J. Bresnan. *The mental representation of gramatical relations*. MIT Press, Cambridge, Massachusetts, 1985.

[CSH02] Robert Corbett, Richard Stallman, and Wilfred Hansen. Bison parser generator, user manual, version 1.35, 2002. (`http://www.gnu.org/software/bison/bison.html`).

[Ear70] J. Earley. An efficient context-free parsing algorithm. In *Communications of the ACM*, volume 13, pages 94–102, 1970.

[GKPS85] G Gazdar, E. Klein, G. Pullum, and I. Sag. *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge, Massachusetts, 1985.

[HKS02] A. Horák, V. Kadlec, and P. Smrž. Enhancing best analysis selection and parser comparison. In *Text, Speech and Dialogue: Proceedings of the 5th International Workshop TSD 2002*, Brno, Czech Republic, 2002. Springer Verlag, Lecture Notes in Artificial Intelligence, Volume 2448.

[Kas65] T. Kasami. An efficient recognition and syntax analysis algorithm for context-free languages. In *Technical report AF CRL-65-758*, Bedford, Massachusetts, 1965. Air Force Cambridge Research Laboratory.

[Kay89a] M. Kay. Algorithm schemata and data structures in syntactic processing. In *Report CSL-80-12*, Palo Alto, California, 1989. Xerox PARC.

[Kay89b] M. Kay. Head driven parsing. In *Proceedings of International Workshop on Parsing Technologies*, Pittsburg, 1989.

[Mea83] Y. Matsumoto and et al. Bup: A bottom-up parser embedded in prolog. In *New Generation Computating*, volume 1, pages 145–158, 1983.

[SH99] P. Smrž and A. Horák. Implementation of efficient and portable parser for Czech. In *Text, Speech and Dialogue: Proceedings of the Second International Workshop TSD'1999*, Pilsen, Czech Republic, 1999. Springer Verlag, Lecture Notes in Computer Science, Volume 1692.

[SH00] P. Smrž and A. Horák. Large scale parsing of Czech. In *Proceedings of Efficiency in Large-Scale Parsing Systems Workshop, COLING'2000*, pages 43–50, Saarbrucken: Universitaet des Saarlandes, 2000.

[SodA93] K. Sikkel and R. op den Akker. Predictive head-corner parsing. In *Proceedings of IWPT'1993*, pages 267–276, Tilburg/Durbuy, 1993.

[SS89] G. Satta and O. Stock. Head-driven bidirectional parsing: A tabular method. In *Proceedings of IWPT'1989*, pages 43–51, Pitsburg, 1989.

[Tom86] M. Tomita. *Efficient Parsing for Natural Languages: A Fast Algorithm for Practical Systems*. Kluwer Academic Publishers, Boston, MA, 1986.

[You67] D.H. Younger. Recognition of context-free languages in time n^3. *Inf. Control*, 10(2):189–208, 1967.