Large Scale Parsing of Czech

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

Abstract

Syntactical analysis of free word order languages poses a big challenge for natural language parsing. In this paper, we describe our approach to feature agreement fulfilment that uses an automatically expanded grammar. We display the insides of the implemented system with its three consecutively produced phases — the core meta-grammar, a generated grammar and an expanded grammar. We present a comparison of parsing with those grammar forms in terms of the parser running time and the number of resulting edges in the chart, and show the need of a shared bank of testing grammars for general parser evaluation.

1 Introduction

Context-Free parsing techniques are well suited to be incorporated into real-world NLP systems for their time efficiency and low memory requirements. However, it is a well-known fact that some natural language phenomena cannot be handled with the context-free grammar (CFG) formalism. Researchers therefore often use the CFG backbone as the core of their grammar formalism and supplement it with context sensitive feature structures (e.g., ?), ?)). The mechanism for the evaluation of feature agreement is usually based on unification. The computation can be either interleaved into the parsing process, or it can be postponed until the resulting structure which captures all the ambiguities in syntax has been built (?).

In our approach, we have explored the possibility of shifting the task of feature agreement fulfilment to the earliest phase of parsing process — the CFG backbone. This technique can lead to a combinatorial expansion of the number of rules, however, as we are going to show in this paper, it does not need to cause serious slow-down of the analysis.

In a certain sense, we investigate the interface between phrasal and functional constraints as described in ?). They compare four different strategies — interleaved pruning, non-interleaved pruning, factored pruning, and factored extraction and see the fundamental asset in the factoring technique. On the other hand, we use a special structure for constraint evaluation. This structure stores all the possible propagated information in one place and allows to solve the functional constraints efficiently at the time of the chart edge closing. Therefore, factoring cannot play such key role in our system.

?) further discussed the possibility of translating the functional constraints to the context-free (CF) phrasal constraints and vice versa and noted that "many functional constraints can in principle be converted to phrasal constraints, although converting all such functional constraints is a bad idea, it can be quite advantageous to convert some of them, namely, those constraints that would enable the CF parser to prune the space of constituents". To date, the correct choice of the functional constraints selected for conversion has been explored mostly for English. However, these results cannot simply be applied in morphologically rich languages like Czech, because of the threat of massive expansion of the number of rules. Our preliminary results in answering this question for Czech suggest that converting the functional constraints to CF rules can be valuable for noun phrases, even if the number of rules generated from one original rule can be up to 56 (see below). An open question remains, how to incorporate the process of expansion to other agreement test checking, especially the subject–predicate agreement and verb subcategorization. Here, the cause of problems are the free word order and discontinuity of constituents omnipresent in Czech. Moreover, ellipses (deletions) interfere with the expansion of verb subcategorization constraints and even of the subject–predicate agreement tests (subject can be totally elided in Czech).

2 Description of the System

We bring into play three successive grammar forms. Human experts work with the meta-grammar form, which encompasses high-level generative constructs that reflect the meta-level natural language phenomena like the word order constraints, and enable 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 and of constructs used to generate combinations of rule elements. The notation of the flags can be illustrated by the following examples:

```
ss -> conj clause
/* budu muset cist -
   I will have to read */
futmod --> VBU VOI VI
/* byl bych byval -
   I would have had */
cpredcondgr ==> VBL VBK VBLL
/* musim se ptat -
   I must ask */
clause ===> VO R VRI
```

The thin short arrow (->) denotes an ordinary CFG transcription. To allow discontinuous constituents, as is needed in Czech syntactic analysis, the long arrow (-->) supplements the right hand side with possible intersegments between each couple of listed elements. The thick long arrow (==>) adds (in addition to filling in the intersegments) the checking of correct enclitics order. This flag is more useful in connection with the order or rhs constructs discussed below. The thick extra-long arrow (===>) provides the completion of the right hand side to form a full clause. It allows the addition of intersegments in the beginning and at the end of the rule, and it also tries to supply the clause with conjunctions, etc.

The global order constraints represent universal simple regulators, which are used to inhibit some combinations of terminals in rules.

```
/* jsem, bych, se -
   am, would, self */
%enclitic   = (VB12, VBK,  R)
/* byl,   cetl, ptal, musel -
   was, read, asked, had to */
%order VBL  = {VL,   VRL,  VOL}
/* byval, cetl, ptal, musel -
   had been, read, asked, had to */
%order VBLL = {VL,   VRL,  VOL}
```

In this example, the %enclitic specifies which terminals should be regarded as enclitics and determines their order in the sentence. The %order constraints guarantee that the terminals VBL and VBLL always go before any of the terminals VL, VRL and VOL.

The main combining constructs in the meta-grammar are order(), rhs() and first(), which are used for generating variants of assortments of given terminals and non-terminals.

```
/* budu se ptat -
   I will ask */
clause ===> order(VBU,R,VRI)
/* ktery ... -
   which ... */
relclause ===> first(relprongr) \
                 rhs(clause)
```

The order() construct generates all possible permutations of its components. The first() and rhs() constructs are employed to implant content of all the right hand sides of specified non-terminal to the rule. The rhs(N) construct generates the possible rewritings of the non-terminal N. The resulting terms are then subject to standard constraints and intersegment insertion. In some cases, one needs to force a certain constituent to be the first non-terminal on the right hand side. The construct `first(N)` ensures that `N` is firmly tied to the beginning and can neither be preceded by an intersegment nor any other construct. In the above example, the `relclause` is transformed to CF rules starting with `relprongr` followed by the right hand sides of the non-terminal `clause` with possible intersegments filled in.

In the current version, we have added two generative constructs and the possibility to define rule templates to simplify the creation and maintenance of the grammar. The first construct is formed by a set of `%list_*` expressions, which automatically produce new rules for a list of the given non-terminals either simply concatenated or separated by comma and co-ordinative conjunctions:

```
/* (nesmim) zapomenout udelat -
   to forget to do */
%list_nocoord vi_list
vi_list -> VI

%list_nocoord_case_number_gender modif
/* velky cerveny -
   big red */
modif -> adjp

/* krute a drsne -
   cruelly and roughly */
%list_coord adv_list
adv_list -> ADV

%list_coord_case_number_gender np
/* krasny pes -
   beautiful dog */
np -> left_modif np
...
```

The endings `*_case`, `*_number_gender` and `*_case_number_gender` denote the kinds of agreements between list constituents. The incorporation of this construct has decreased the number of rules by approximately 15%.

A significant portion of the grammar is made up by the verb group rules. Therefore we have been seeking for an instrument that would catch frequent repetitive constructions in verb groups. The obtained addition is the `%group` keyword illustrated by the following example:

```
%group verb={
    V:head($1,intr)
      add_verb($1),
    VR R:head($1,intr)
         add_verb($1)
         set_R($2)
}

/* ctu - I am reading */
/* ptam se - I am asking */
clause ====> order(group(verb),vi_list)
```

Here, the group `verb` denotes two sets of non-terminals with the corresponding actions that are then substituted for the expression `group(verb)` on the right hand side of the `clause` non-terminal.

Apart from the common generative constructs, the metagrammar comprises feature tagging actions that specify certain local aspects of the denoted (non-)terminal. One of these actions is the specification of the head-dependent relations in the rule — the `head()` construct:

```
/* prvni clanek - first article */
np -> left_modif np
  head($2,$1)
/* treba - perhaps */
part -> PART
    head(root,$1)
```

In the first rule, `head($2,$1)` says that (the head of) `left_modif` depends on (the head of) `np` on the right hand side. In the second example, `head(root,$1)` links the `PART` terminal to the root of the resulting dependency tree. More sophisticated constructs of this kind are the `set_local_root()` and `head_of()`, whose usage is demonstrated in the following example:

```
/* ktery ... -
   which ... */
relclause ===> first(relprongr) \
                 rhs(clause)
    set_local_root(head_of($2))
```

Here, the heads in `rhs(clause)` are assigned as specified in the derivation rules for `clause`. This way we obtain one head of the `rhs(clause)` part and can link all yet unlinked terms to this head.

The Second Grammar Form (G2) As we have mentioned earlier, several pre-defined grammatical tests and procedures are used in the description of context actions associated with each grammatical rule of the system. We use the following tests:

- grammatical case test for particular words and noun groups

```
noun-gen-group  ->  noun-group \
                      noun-group
   test_genitive($2)
   propagate_all($1)
```

- agreement test of case in prepositional construction

```
prep-group  ->  PREP \
                  noun-group
   agree_case_and_propagate($1,$2)
```

- agreement test of number and gender for relative pronouns

```
ng-with-rel-pron  ->  noun-group \
                  ',' rel-pron-group
   agree_number_gender\
     _and_propagate($1,$3)
```

- agreement test of case, number and gender for noun groups

```
adj-ng  ->  adj-group noun-group
   agree_case_number_gender\
     _and_propagate($1,$2)
```

- test of agreement between subject and predicate
- test of the verb valencies

```
clause  ->  subj-part verb-part
   agree_subj_pred($1,$2)
   test_valency_of($2)
```

The contextual actions `propagate_all` and `agree_*_and_propagate` propagate all relevant grammatical information from the nonterminals on the right hand side to the one on the left hand side of the rule.

Expanded Grammar Form (G3) The feature agreement tests can be transformed into the 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.

2.1 Parser

In our work, we have successively tried several different techniques for syntactic analysis. We have tested the top-down and bottom-up variants of the standard chart parser. For more efficient natural language analysis, several researchers have suggested the concept of head-driven parsing (e.g., ?), ?)). Taking advantage of the fact that the head-dependent relations are specified in every rule of our grammar to enable the dependency graph output, the head-driven approach has been successfully adopted in our system. Currently, we are testing the possibility of incorporating the Tomita's GLR parser (?; ?) for the sake of comparing the efficiency of the parsers and the feasibility of implanting a probabilistic control over the parsing process to the parser.

Since the number of rules that we need to work with is fairly big (tens of thousands), we need efficient structures to store the parsing process state. The standard chart parser implementation used in our experiments employs 4 hash structures — one for open edges, one for closed edges, one hash table for the grammar rules (needed in the prediction phase) and one for all edges in the agenda or in the chart (the hash key is made of all the attributes of an edge — the rule, the dot position and the surface range). In the case of a head-driven chart parser, we need two hashes for open edges and also two hashes for closed edges.

The gain of this rather complex structure is the linear dependency of the analysis speed on the number of edges in the resulting chart. Each edge is taken into consideration twice — when it is inserted into the agenda and when it is inserted into the chart. The overall complexity is therefore $2k$, where k is the number of edges in the resulting chart.

The number of chart edges that are involved in the appropriate output derivation structure is related to:

a) the number of words in the input sentence, and

b) the ambiguity rate of the sentence.

The output of the chart parser is presented in the form of a packed shared forest, which is also a standard product of the generalized LR parser. Thus, it enables the parser to run the postprocessing actions on a uniform platform for the different parsers involved.

During the process of design and implementation of our system, we started to distinguish four kinds of contextual actions, tests or constraints:

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

Rule-tied actions are quite rare and serve only as special counters for rule-based probability estimation or as rule parameterization modifiers. Agreement fulfilment constraints are used in generating the G3 expanded grammar, in G2 they serve as chart pruning actions. In terms of (**?**), the agreement fulfilment 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. They are used, for instance, in the packed dependency graph generation. On the other hand, there are some actions that do not need to work with the whole chart structure, they are run after the best or n most probable derivation trees are selected. These actions do not prune anything, they may be used, for example, for outputting the verb valencies from the input sentence.

3 Results

In our system, we work with a grammar of the Czech language (**?**), which is being developed in parallel with the parsing mechanism. The grammar in the three forms, as exemplified above, has the following numbers of rules:

G1 meta-grammar – # rules	326
G2 generated grammar – # rules	2919
shift/reduce conflicts	48833
reduce/reduce conflicts	5067
G3 expanded grammar – # rules	10207

As a measure of the ambiguity rate of G2, we display the number of shift/reduce and reduce/reduce conflicts as counted with a standard LR parser generator. These data, together with the number of rules in the grammar, provide basic characteristics of the complexity of analysis.

The comparison of parsing times when using the grammars G3 and G2 without the actions taken into account is summarized in Table 1. We present the time taken for parsing a selected subset of testing sentences — only sentences with more than 50 words were chosen.

The results show that in some cases, which are not so rare in highly inflectional languages, the expanded grammar achieves even less number of edges in the chart than the original grammar. This effect significantly depends on the ambiguity rate of the input text. A question remains, how to exactly characterize the relation between ambiguity in the grammar and in the input.

The fully expanded grammar G3 is only moderately larger than the G2 grammar (about three times the size). The reason lies in the fact that the full expansion takes place mainly in the part of the grammar that describes noun phrases. This part forms only a small amount of the total number of G2 rules. Considering this, it is not surprising that the parse times are not much worse or even better. It also benefits from early pruning by transforming the unification constraints into the CFG. The agreement tests between subject and predicate should also be expanded. Nevertheless, we do not do this, since the position of subject is free, it cannot be described with CF rules without imposing a huge amount of ambiguity to every input sentence.

4 Packed Dependency Graph

Ambiguity is a fundamental property of natural languages. Perhaps the most oppressive case of ambiguity manifests itself on the syntactic level of analysis. In order to face up to the high number of obtained derivation trees, we define a sort

Sent	**# of**	**G2**		**G3**		**# edges**
#	**words**	**# edges**	**time**	**# edges**	**time**	**G3/G2**
1100	52	115093	1.95	143044	2.53	124 %
1102	51	84960	1.56	107318	2.02	126 %
1654	51	202678	3.23	361072	6.81	178 %
1672	59	269458	4.08	434430	7.13	161 %
1782	51	212695	3.36	168118	3.05	79 %
2079	66	98363	1.83	223063	3.75	227 %
2300	60	262157	4.03	443022	7.28	169 %
2306	102	739351	12.94	715835	10.95	97 %
2336	103	355749	5.21	565506	8.83	159 %

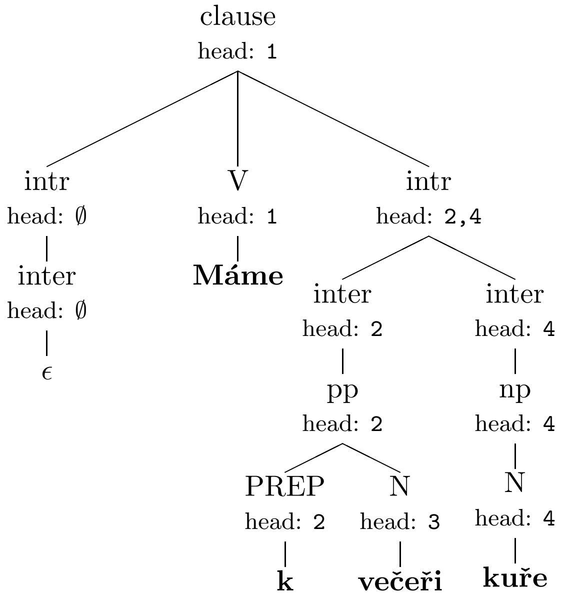
Table 1: Running times for G2 and G3

order on the output trees that is specified by probabilities computed from appropriate edges in the chart structure. The statistics is also involved in the process of sorting out the edges from the agenda in the order that leads directly to n most probable analyses.

A common approach to acquiring the statistical data for the analysis of syntax employs learning the values from a fully tagged tree-bank training corpus. Building such corpora is a tedious and expensive task and it requires a team cooperation of linguists and computer scientists. At present, the only source of Czech tree-bank data is the Prague Dependency Tree-Bank (PDTB) (**?**), which contains dependency analyses of about 20000 Czech sentences.

The linguistic tradition of Czech syntactic analysis is constituted by distinguishing the role of head and dependent and describes the relations between a head and its dependents in terms of semantically motivated dependency relations. In order to be able to exploit the data from PDTB, we have supplemented our grammar with the dependency specification for constituents. Thus, the output of the analysis can be presented in the form of a pure dependency tree. At the same time, we unify classes of derivation trees that correspond to one dependency structure. We then define a canonical form of the derivation to select one representative of the class which is used for assigning the edge probabilities.

The dependency structures for all possible analyses are stored in the form of a packed dependency graph. Every "non-simple" rule (that has more than one term on the right hand side) is extended by a denotation of the head element and its dependents. Thus, the dependency is often given as a relation between non-terminals, which cover several input words. However, the basic units of the dependency representation are particular surface elements (words). To be able to capture the standard dependency relations, we propagate the information about a "local head" from the surface level through all the processed chart edges up to the top. A simplified case that captures only one possible derivation of sentence '`Máme k večeři kuře.`' (*We have a chicken for dinner.*) can be described by the following tree:



During the evaluation of post-processing actions, every head-dependent relation is then recorded as an edge in the graph (without allowing multi-edges). An example of the graph for

Figure 1: Dependency graph.

Figure 2: Two of the four possible dependency trees.

the sentence 'Však Martin s koněm bydlel u strýce se špinavým prasetem.' (literally: *However, Martin with a horse lived with his uncle with a dirty pig.*) is depicted in Figure 1. Two examples of unpacked derivation trees that are generated from the graph are illustrated in Figure 2.

The packed dependency graph enables us to recover all the possible standard dependency trees with some additional information gathered during the analysis. The example graph represents two dependency trees only, however, in the case of more complex sentences, especially those with complicated noun phrases, the saving is much higher.

5 Conclusions

In this paper, we have shown that shifting all possible feature agreement computations to the CFG backbone is suitable for free word order languages and it does not need to cause a serious increase in parsing time. We discuss three consecutively produced forms of our grammar and give a comparison of different parser running times on highly ambiguous input.

In the process of parsers evaluation, we lacked the possibility to compare the parsing efficiency on a large number of testing grammars. These grammars cannot be automatically generated, since they should reflect the situation in real-world parsing systems. Future cooperation in NL parsing could therefore lead to the creation of a commonly shared bank of testing grammars with precisely specified ambiguity measures.

References