The Prague Bulletin of Mathematical Linguistics 77, 2002

On the Rule-Based Parsing of Czech

Kiril Ribarov

This paper presents various attempts to accustom the Rule-based Approach (RBA) as originally introduced by Eric Brill in 1993 (Brill 1993a), on the problem of parsing of Czech (a highly inflective language) within a dependency based syntactic framework (Sgall et al 1986). It is experimentally supported in this paper that the modification of RBA for the stated aim is neither simple nor straightforward for all attempts fail to produce such a set of rules that parses Czech with a comparable success rate to the one obtained for English or the one obtained for the currently best parser (statistical) for Czech (Collins et al. 1999).

At the beginning of this article the parsing problem restricted to the need of this work is specified. Further, the results of the first attempts to modify the rules for RBA parsing of Czech is presented, the influence of the cardinality of the tag set size of the parsing success rates is studied and a second attempt at a wider modification of the rule templates is provided. The article contains the basic accompanying experimental results and aims to be a solid background and a starting point for any adaptation of the rule-based approach for the purposes of parsing of an inflective language within a dependency based framework.

1. On the understanding of parsing

1.1. Firstly

The question what a parser is can have a very trivial but at the same time a rather complex answer. To start with, I use the "trivial" one stating that a parser is a process of assigning a (labeled or unlabeled) syntactic tree structure to a sentence at the input; the input originally does not exhibit any kind of syntactic information. By a syntactic tree structure (being assigned during the parsing) I denote such syntactic structures as those being present in the treebank in use (PDT 2001).

Let IN be an input sentence and SYN_OUT be the same sentence enriched with its syntactic information. Parsing can be defined as an algorithm transforming IN into SYN_OUT.

Therefore, parsing can be viewed as a mapping between sets of INs (S) and SYN_OUTs (**S**):

$$\textit{parsing}: S \longrightarrow \mathbf{S}.$$

The desired mapping to be performed has to be classified as a very sparse one. To support this claim the number of combinatorially possible structures is given in the second column of Table 1 (if, as in the case of (Brill 1993b), only complexes of more than one member are understood as constituents) as opposed to linguistically located structures in the Penn Treebank in the third column (the combinatorial space of a dependency tree is even larger). Their ratio is presented in the fourth column. Assuming that a sentence of 20 tokens (including punctuation) is very likely, the ratio of found structures over all possible structures is of a value of 2.4×10^{-8}. An algorithm using no linguistic knowledge will have to test all of those possibilities before selecting the correct parsing. From this perspective, the algorithm of automatic grammar generation could be specified as an algorithm of effective linguistic knowledge search within an extremely sparse combinatorial space.

i: sentence length	T(i) = (i-1)T(i-2) + T(i-1) possible structures[1]	Found structures	Ratio
0	1		
1	1		
2	1		
3	3		
4	6		
5	18		
6	48	25	0,52083333
7	156	24	0,15384615
8	492	40	0,08130081
9	1740	65	0,03735632
10	6168	70	0,0113489
11	23568		
12	91416		
13	374232		
14	1562640		
15	6801888		
16	30241488		
17	139071696		
18	653176992		
19	3156467520		
20	15566830368	379	2,4347E-08

Table 1: The Combinatorial Sparseness of Parsing

The most traditional understanding of parsing would be such where *parsing* is a mapping which assigns all possible syntactic structures to the input. This is inherent mainly to hand crafted systems (e.g. manually designed formal grammar parsers). As for the automatic procedures *parsing* is not just a mapping but a function, a single structure is assigned to a single input sentence. Although statistical parsing algorithms can be programmed to output not only the most probable output but also the less probable ones, thus giving more than a single output (within the rule-based approach it is more difficult to make such modifications) it is assumed that statistical and rule-based techniques (and other of similar nature) are not capable of a complete parsing of the input sentence. I have in mind that parsing primarily denotes an analysis according to the traditional understanding; nevertheless I will use the term parsing (and parser) for automatic procedures for syntactic analysis at any level of the syntactic description. These automatic procedures, as noted above, give mainly a single syntactic analysis of the input stream, although the input might exhibit more than one syntactic structure.

As implicitly mentioned, the structure of the output 'mimics' the structure present in the treebank. The assignment of a parse to a sentence may be viewed in two steps:

(a) assigning a tree structure to the sentence, and

(b) assigning syntactic attributes to the sentence (dependency) elements.

Three scenarios are possible: first (a) then (b), first (b) then (a), or (a) and (b) together.

More freedom could be given to the discrete representation of the syntactic structures which are the output of the parser: assigning a syntactic structure to the input does not necessarily have to mean

[1] The series represents the possible binary tree structures algorithmically possible, as only such are produced from Brill's rule-based parser. The total number of the possible tree structure is represented by the Catalan numbers series.

assigning only a tree structure. But if a non-tree structure is assigned, it should be transformable to a syntactic tree structure without loss of information.

The syntactic structure to obtain is a surface syntactic structure (analytical structure)[2] as defined and used in the Prague Dependency Treebank (PDT). A parser is the algorithm implementing the *parsing* as understood above.

1.2. Secondly

Revealing the analytical structure of the sentence is a complex process. Various experiments conveyed allow us to view syntactic analysis as a system with several modules (Figure 1).

The parser is the main module of such a system, with the following requirements for its input and output:

The input: A sentence or a word group acting as a sentence (e.g. a table). The sentence can be associated with its morphological information (see later on modifications of this information and selection of morphological information relevant for syntactic parsing).

The output: analytical syntactic structure of the input sentence, or the input sentence accompanied by syntactic information such that it can be easily transformed into an analytical tree structure of the input sentence, or into a partial analytical tree that provides at least as much information as an analytical tree would have provided for the step to come, the deep syntactic parser (which is not a part of this work) representing the tectogrammatical (underlying syntactic) structure of the sentence.

As stated earlier, a proper output should give all possible parses of the input sentence. Unfortunately, this aim is not possible to fulfil with the methods used here, although modifications of the automatic procedures that would not exclude different parses of the sentence (or make them impossible) are under development.

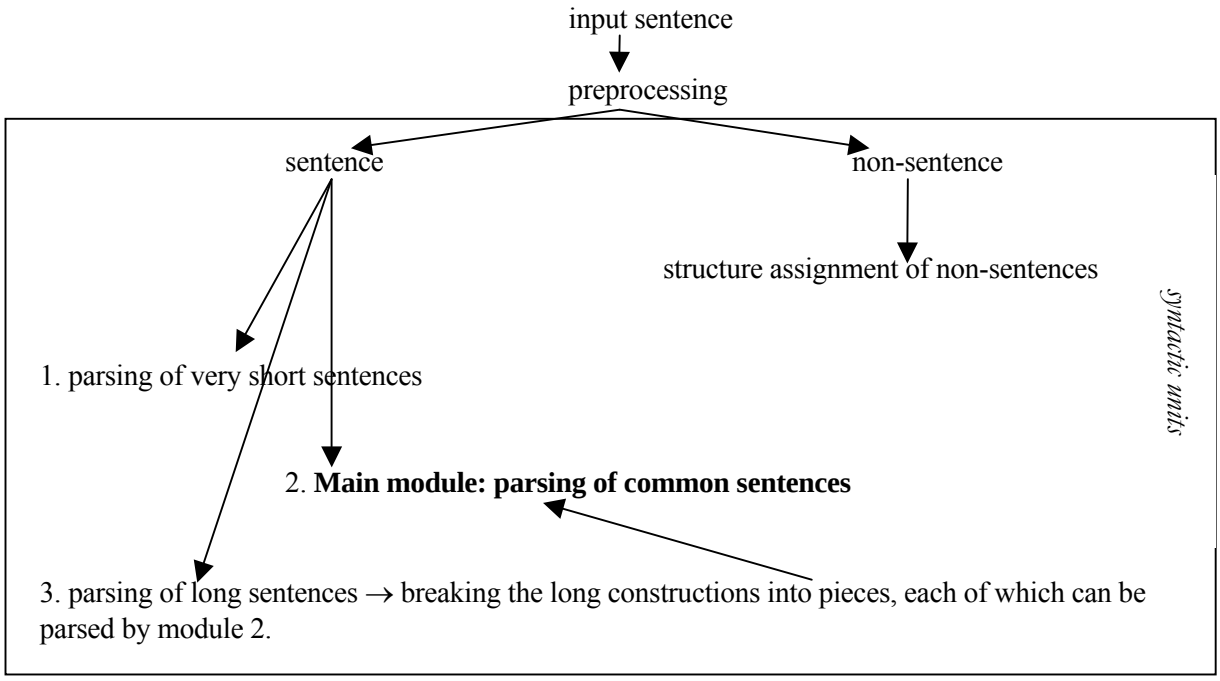

Figure 1: Parsing components

Besides the main module, other parts of the system are briefly described in the sequel.

[2] Currently, procedures for automatic transformation of analytical (surface syntax) trees to tectogrammatical (deep syntax) ones is under development (Böhmova 2001).

The process of preprocessing identifies any kind of regular and fixed information. Its aim is to localize the regular parts of the text and the regular parts of the sentences, such as:

- identification of tables, lists, sport event results (taken to be non-sentences)

- localization of very short and very long sentences (sentences with a large number of verbs and rich punctuation)

- identification of compound verbs (verbal analytical forms)

- identification of prepositions and prepositional groups

- certain coordinations

- certain parenthetical parts

- certain idioms and phrases.

This list can be modified and is dependent on the language being processed. As mentioned earlier, the Czech language is considered here. The criterion of what can be a part of the preprocessing module is given by the ability of designing preprocessing rules such that they will not reduce the recall of the system. The need to separate long sentences from shorter ones results from unambiguous experiments, which show that the longer the sentence is the less successful is the parser. Breaking long sentences into smaller sentential segments reduces the combinatorial space of possible structures to be searched.

Having in mind that PDT consists of various texts most of which are newspaper texts included in PDT without any editorial work or selections, one must expect a variety of non-sentences in the input. These non-sentences, like tables, lists, sport or chess results etc., are not our main objects of interest. Although being assigned an analytical structure, such a segment is dependent on the spelling norms of the language and exhibits firm regularities. Therefore, it is believed that if they are recognized and located correctly, specialized modules based on regular expressions can process them. It is our aim, if not to process them, at least to recognize them.

During the whole process, verification of a 'list of language truths', called syntactic units, can be continuously performed**.** Syntactic units are language dependent truths about the processed language. As for Czech these units are to be searched in:

- analytical forms (usually verbal; it is also a part of the preprocessing)

- check on morphological agreement

- list of atypical syntactic constructions based on the manual for analytical annotation, where such structures are exemplified, taken as a list of structured exceptions

- results of a study of the discrete nature of the tree structures present in the treebank; non-linguistic information that refers to the tree structures and their dimensions in the treebank.

An ideal syntactic unit is such that does not reduce the recall of the system while not decreasing its precision. For a practical use a syntactic unit is such that behaves as an ideal one within certain pre-stated threshold values. Syntactic units can be applied continuously throughout the process of parsing or in certain cycles as a post-processing (correction) unit. Syntactic units can also be a part of a preprocessing unit. The way how syntactic units are applied influences the success rate of the system.

Speaking of parsing very short sentences I have in mind parsing sentences without verbs. In such sentences one may expect incomplete grammatical constructions having originally other than basic declarative, exclamatory or interrogative character in the text. Usually headers or other titles have these characteristics. It is justifiable to assume that such sentences can be successfully localized, which would allow, if needed, to process them separately (e.g. the rule-based method specially trained on such sentences could parse them in a satisfactory manner).

The main (parsing) module is independent on the type of the parser being used, but the system as designed expects an automatically trained parser being either a rule-based one or a statistical one. Any combinations of both are not excluded.

To avoid misinterpretations and false expectations, I would like to stress that the parser under consideration is such that is automatically trained to recognize the syntax of the language, and does not include any human interactions; by rules I mean automatically learned rules, rather than manually crafted ones.

2. The first modifications of the rule-based approach for parsing of Czech

In my master thesis (Ribarov 1996) and as later published in Hajič, Ribarov (1997), for the first time an attempt at an application of a rule-based approach (originally proposed by Brill) for an automatic grammar generation of Czech was demonstrated. The mentioned work grew in a time when no treebank of Czech was available; a small data bank had to be created to allow for testing the algorithm and building and analyzing its modifications in order to be able to apply it for a dependency syntactic framework of Czech. Since these results are important for the later development and improvements, I summarize here the results of the above mentioned contributions. At the same time I use this occasion to introduce the basics of the rule-based approach and the used terminology.

2.1. The Rule-Based Approach

Belonging to the group of algorithms for automatic natural language learning the *rule-based approach* applied to *parsing* can be summarized in the following diagrams, one for the process of application used to parse an input text when a set of rules exists (Figure 2) and the other one for the process of learning during which the *set of rules* is created (Figure 3).

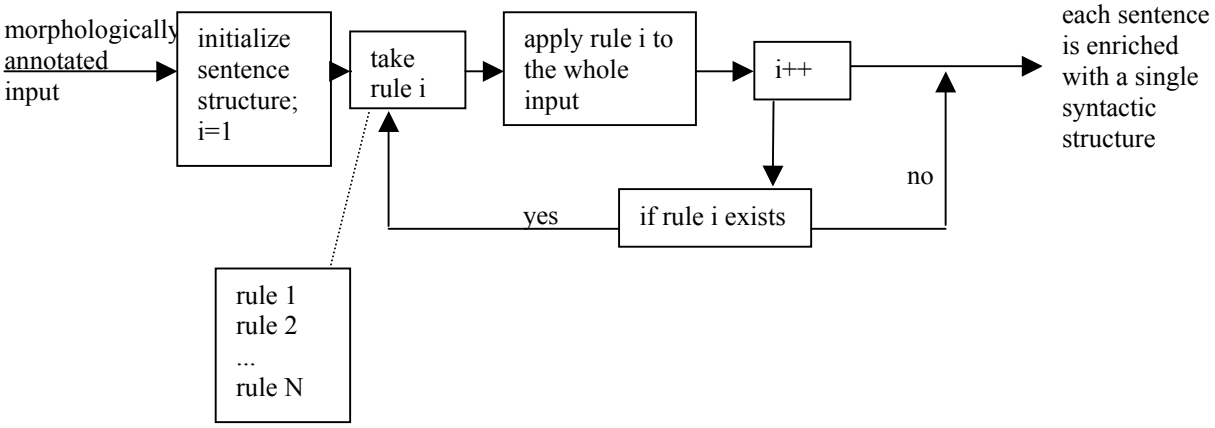

Figure 2: The rule-based approach: application of rules

The rules are obtained automatically, although one may imagine also a set of manually written rules being a part of such a process[3]. The real power of the rule-based approach is in the ability to learn the rules from a syntactically preprocessed input. The supervised learning is performed on a rather small set[4] of manually syntactically processed sentences. Manually in this sense means correctly syntactically analyzed sentences. The set of manually analyzed sentences used for the process of learning is usually referred to as the *truth*.

[3] It is possible to add manually designed rules to the set of automatically obtained rules, but this is not the aim of this study.
[4] Opposed to the statistical techniques, the rule-based learning procedure requires less data.

Before one starts to apply the rule-based approach, *templates* of the rules must be known (given externally to the system) and a procedure, a function that will decide how well a specific instance of a rule performs, i.e. the comparison function between the truth and the actually modified input, known as the *error function*, must be known. The learning can then be schematized as in Figure 3.

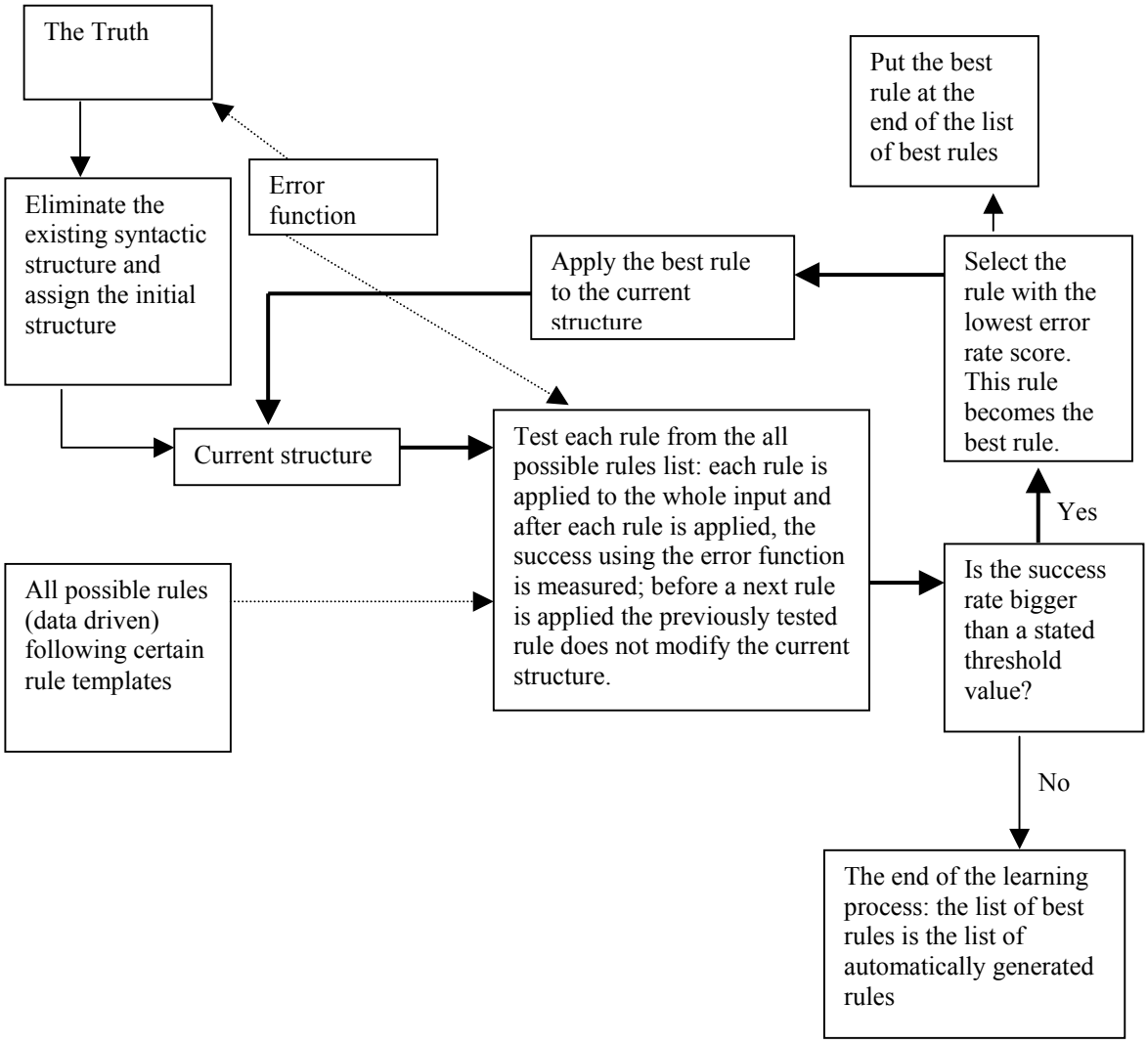

Figure 3: The rule-based error driven learning - the automatic learning of the rules

There are the following connections between Figure 2 and Figure 3:

- The initial syntactic structure during learning should be the same as the initial syntactic structure during application.
- The error function used while learning is usually, but not necessarily, the function used during evaluation.
- If, as shown in Figure 2, the input is enriched with additional information e.g. morphological, than the same enriched input needs to be present also during the process of learning.
- The list of best rules obtained in the process of learning is the same list of rules used during the process of application, preserving the very order in which they have been obtained.
- Each rule has the following formal structure:

```
<action> <what> <where>
```

where <action> is the action to be performed, usually addition, change or deletion of certain elements; <what> determines the element on which the action is performed (for the parsing it is either a constituency boundary or a dependency), and <where> is the definition of the (triggering) environment on which the action is performed. For a rule to be performed the data as conditioned in <where> together with the element specified in <what> must be located and the action specified in <action> must be possible with respect to <what> within <where>. E.g. DELETE LEFT_PAREN[5] BETWEEN NOUN NOUN means: delete a left constituency boundary between two neighboring nouns. In order to perform this rule two neighboring nouns must exist (selection of the environment) and a LEFT_PAREN between the two nouns must exist; otherwise there is nothing to delete.

After a rule is performed, technical adjustments of the modified structure can be performed, the aim of which is to maintain a connected, coherent syntactic tree structure: deleting a left constituency boundary cannot be left alone without deleting the matching right constituency boundary; further, the syntactic tree after such a modification is not necessarily a tree anymore, therefore maintaining steps are performed in order to keep the syntactic structure as a tree structure; such steps can be of various types.

For further information on how the rule-based approach is applied to syntactic parsing of English within a constituency based framework, or how the rule-based paradigm is applied to PP attachment, or to POS tagging, one is advised to consult Brill (1993a); Brill (1993b); Brill (1992).

2.2. Modifications for Czech

Originally introduced and explained only for the constituency grammatical framework, in order to be applied for a dependency framework, as mentioned earlier, the rule-based approach needed to undergo deep changes, which are most visible principally in two of its components: the rule templates and the error function. A closer and more detailed elaboration of the changes will be given in Section 4. Here I elaborate the very first modifications made, the logic of which is present also in the sequential work.

The following rule templates which operate on the linear description of the sentence and assign only the bare syntactic tree structure (not the syntactic functions) have been proposed:

Rules	**Explanation**
SWAP LPAREN BETWEEN A B	Swap A and B if there is a left bracket in-between them
SWAP COMMA BETWEEN A B	Swap A and B if there is a comma in-between them
ADD LPAREN BETWEEN A B	Add left bracket between A and B
DEL LPAREN BETWEEN A B	Delete left bracket between A and B if there is any

Table 2. Dependency rules[6]

For clarification and for better understanding of Table 2 and of the term bracket, see Figure 4, which displays the linear description accompanied by its unambiguous dependency tree structure:

A (B, C(D, E), F(G))

[5] By LEFT_PAREN or LPAREN or '(', or RIGHT_PAREN or RPAREN or ')' I refer to left parenthesis or right parenthesis, respectively.

[6] Four other rules were also considered dealing with apposition and coordination: ADD/DEL APOS/COORD BETWEEN A B. These rules added a new node to the tree such that joined two coordinated subtrees or establish a relation between two subtrees in apposition. In the analytical syntactic anotation of PDT no extra nodes can be added, coordination and apposition became syntactic attributes, analytical function values, assigned to a certain lexical item representing the relation of coordination or apposition in the tree structure as the conj. "and" can be a coordinating node for Peter and Mary in e.g. "Peter and Mary bought a present".

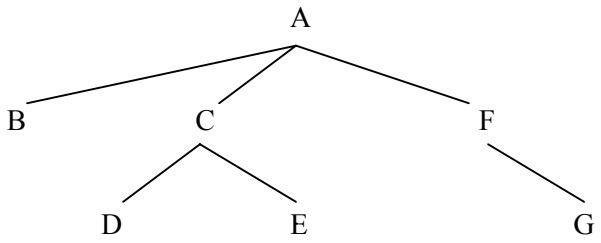

Figure 4: Example of a tree structure

Each left bracket has its "nominated" governor: e.g. A is a governor for the left bracket between A and B, or C is a governor for the left bracket between C and D, etc.

In comparison with the rules used for obtaining immediate constituents[7] (Brill 1993b), the rules in Table 2 imply the following major differences:

- They delete brackets, without adding new ones.
- They add new brackets, without deleting some previous ones.
- They change the word order in a sentence.

Table 3 presents the conditions under which the modified rules are defined.

Rule	Defined if:
SWAP LPAREN BETWEEN A B DEL LPAREN BETWEEN A B	There is only one left bracket between A and B (that is: B is the leftmost son of A)
SWAP COMMA BETWEEN A B ADD LPAREN BETWEEN A B	There is no left bracket between A and B (that is: A and B have the same governor and B is next to A)

Table 3: Rule conditions

If one manipulates with a node, then this has an influence on its subtree. The rules containing SWAP change the order of words in a sentence. This has a direct influence on the possible triggering environments, which change after each selection of the best rule within the RBEDL process, due to the fact that the surface word order is not preserved[8].

An application of each of the rules specified above results in the following modifications (see Fig. 3 to Fig. 6) as exemplified on the original simplified tree structure A(B(C,D),E) given in Fig. 2.

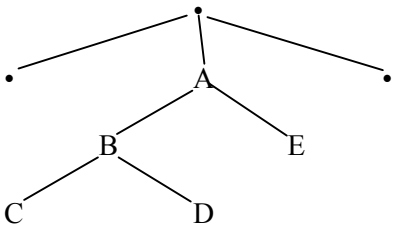

Figure 5. Simplified tree structure A(B(C,D),E)

[7] For the lack of space those are not included here.
[8] Within the approach based on constituency structure the word order is stable. With our framework, possible changes between the surface word order and the order of items in our linear tree representations significantly increase the complexity of the learning process.

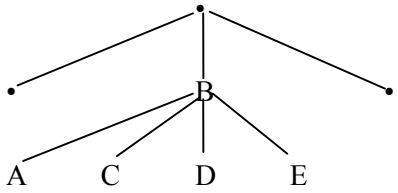

Figure 6. SWAP LPAREN BETWEEN A B

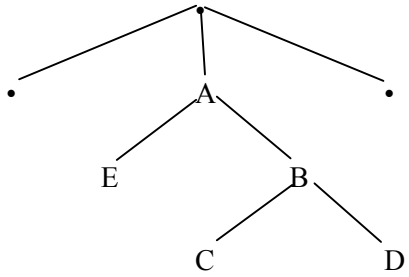

Figure 7. SWAP COMMA BETWEEN B E

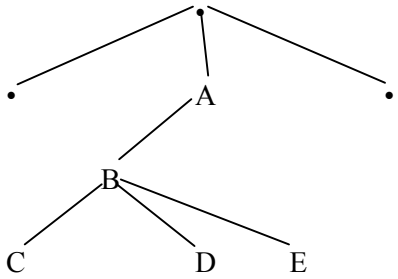

Figure 8. ADD LPAREN BETWEEN B E

The error function is completely different from the one used for obtaining the phrase structure that counts the crossing constituents, therefore a direct comparison of the success rates between immediate constituent structure and dependency structure is not possible. A function sensitive to differences of the syntactic structure considered as important was created. It is not advisable to have an error function that would be too sensitive for certain types of changes and unequally insensitive for other types of differences in the tree structure. The error function counts the dependency relations present in the tree with respect to the total number of correct dependencies.

For the tree structure A (B, C(D (E)), F), where the capital Latin letters stand for a tag or a word, there are 5 dependencies as follows:

A(B, A(C, A(F, C(D, D(E.

Let us compare the above analysis to another tree structure of the same sentence:

A (B (C (D, E)), F),

which has the following five dependencies:

A(B, A(F, B(C, C(D, C(E.

The two structures have the following three dependencies in common:

A(B, A(F, C(D,

therefore if one of the structures is 'correct' the success rate is 3/5.

In order to initiate the process of learning and/or the process of parsing the input sentence to be processed is assigned an initial tree structure. The left chain structure has been chosen, as demonstrated on the following examples, to be the initial tree structure:

(A), for a sentence of length 1[9], or

A(B), for a sentence of length 2, or

A (B , C), for a sentence of length 3, or

A (B (C), D), for a sentence of length 4, or

A (B (C , D) , E), for a sentence of length 5, and so on.

2.3. Results: summary and considerations

The rule-based error driven learning is sensitive on different parameters present in the data or set in the learning process (length of sentences, initial condition) which indirectly influence the results. The tag set size and its structure is one of the major factors (for Czech, the full tagset size is approx. 3500 tags) that influence the success rate: the smaller the morphological tagset is the more successful is the learning process. However, for syntactic analysis, the tag set does not need to include all morphological features (Ribarov, 1996). Therefore, after the tag set had been restricted (while still being larger than the tag set of the Penn Treebank), some small size experiments were carried out. At the beginning (when there was no treebank and there was a need of testing the pure method as such), different very small training sets were created, which contained approx. 25 sentences (the rule-based parser for Czech, without further improvements, does not perform significantly better if trained on more than 100 sentences, see Section 4.3). The average success rate of initial bracketing was estimated to be 42%. Various sets of rules were trained (each set for a different learning set) consisting of approx. 27 rules learned. The rules were tested on a set of 20 sentences. Each of the set of rules increased the success of bracketing by approx. 8.5%. Those were the very first steps: far from impressive results and astonishing percentages, but still not demotivating ones. The rules were rather simple, and the success rate for the dependency structure is always more pessimistic than e.g. that of crossing constituents in the immediate constituents structure, and the improvement of 8.5% percent was judged to be relatively not a bad improvement. It was clear that it could be amended in various aspects. Many changes were made in-between, both in the data and in the modifications of the stated approach. Everything has been done with the primary aim to judge the strength and the potential of the automatically generated rules, guided by a (secondary but more important) aim of constructing a suitable parser for Czech.

3. Towards different modifications of the RBA for parsing of Czech

After the first attempt to apply the rule-based approach for a dependency based syntactic analysis of Czech several other steps and modifications were made. Most important of all, the modified RBA was tested on real corpus data, as soon as those were available. An increasement of the success rate was observed, higher than the one in the early experiments from Subsection 2.3. However, the success rate has not yet been high enough for any sensible automatic application. To reach a higher success rate, several other modifications were made: the set of rules was changed and enriched with new rules, another error rate function was considered, various morphemic tag sets were tested. The present section

[9] A corpus might contain sentences represented only by a full stop as it was in our data.

contains relevant information for approaches of an application of the rule-based approach for parsing Czech[10].

3.1. The input

In all of the experiments, the input text is taken from the Prague Dependency Treebank (PDT 2001), and the rules operate on the morphemic representation of the input text. The sentence:

Počet kopií z jedné kazety se pohybuje kolem 9 až 10 tisíc.

[The number of copies from one cartridge is about 9 to 10 thousand.]

has the following morphemic categories[11]:

Počet|NNIS1-----A----		number
kopií|NNFP2-----A----		copies
z|RR--2----------	from	
jedné|CIFS2----------		one
kazety|NNFS2-----A----		cartridge
se|P7-X4----------	is-*refl.*	
pohybuje|VB-S---3P-AA---		move
kolem|RR--2----------		about
9|C=----------		9
až|J,----------		to
10|C=----------	10	
tisíc|CIXS2----------		thousand
.|Z:----------	.	

The input rules ignore the lexical items and take the sentence to parse to be the input stream of morphemic tags

NNIS1-----A---- NNFP2-----A---- RR--2---------- CIFS2---------- NNFS2-----A---- P7-X4---------- VB-S---3P-AA--- RR--2---------- C=---------- J,---------- C=---------- CIXS2---------- Z:----------

The dependency tree is represented in its linear form, where S is the sentence marker, an artificially introduced node governing the whole sentence.

S(pohybuje(Počet(kopií(z(kazety(jedné)))), se, kolem(tisíc(až(9, 10)))), .)[12]

In the earlier stages when no sufficiently large disambiguated morphological data were available, but one had only the output of a complete morphological analysis of the input words, there was a need of accustoming the algorithms to that situation. It was then necessary to state where it was possible to apply the readings/interpretation of the rules such that the non-disambiguated morphemic tags could be used.

Rules	**Explanation**
SWAP LPAREN BETWEEN A B SWAP COMMA BETWEEN A B ADD LPAREN BETWEEN A B DEL LPAREN BETWEEN A B	All of the rules are read in the same way as before assuming that A and B are not disambiguated tags but members of a set of possible morphemic tags.

Table 4: Explanation of modified basic rule types

[10] Several experiments for tagging Czech are also included.
[11] For information on the positional tagset for Czech, see (PDT 2001).
[12] For reasons of clarity the linearized description of the syntactic structure of the sentence includes only the lexical items instead of the morphemic categories.

The present computational power and the combinatorial character of the learning process do not allow a practical use of big tag sets and do not allow more than one tag per word. Therefore the requirement is to have a disambiguated morphology and to operate on disambiguated tags[13], at least for the process of learning (while during application one could assume input with multiple tags and interpret the rules as in Table 4).

3.2. The influence of modified tagsets[14]

Assuming that the input sentence to parse consists only of morphemic tags, the type and the size of the tagset has a direct influence on the success rate of the parsing procedure. The motivation for this study can be found in the following:

- The success of tagging of Czech is influenced by the cardinality of the tagset. As discussed in Hladká and Ribarov (1998) and also supported by a former study (Elworthy 1995), where experiments concerning changing tagsets for three different languages (English, French, Swedish) are presented, the tagset should be chosen according to the requirements of a given application. The aim is to combine a morphemic tagger and a parser in order to obtain a fully automatic procedure. Therefore the output of the tagger should, in our case, suit best the needs of the parser.
- Besides that, in order to combine a tagger and a parser one needs to have a tagger with the highest possible success rate (the parser should then be trained on the output of the tagger).

Practically this is realized by examining the performance of the parser on tagsets of different sizes. At the same time the success rate of the tagger itself for different tagsets should be observed, in order to select the most successful one. Three different POS tagsets (POS TAG_{1171}, POS TAG_{206}, POS TAG_{34})[15], a statistical approach to tag text and the tagging accuracy of each statistical tagging with three different tagsets have been developed and tested. The results are summarized in Table 5.

Method[16]	BMM	RB	BMM	RB	MMFS	MMFS	MMFS
POS tagset size	1 171	1 171	206	206	47	43	34
Training data	600K	38K	600K	38K	15K	15K	15K
Tagger accuracy	81,5%	79,8%	90,1%	87,2%	91,7%	93,0%	96,2%

Table 5: Tests on various tagsets

For a user, let us say a linguist, whose aim is to specify morphological categories, we need to perform tagging with a full tagset, more specifically, with carefully selected and detailed morphological classification corresponding to the traditional grammar. The result could be viewed as a process which „added“ information to the text, or as a way of classification (clustering) of the word stock. Different tagset sizes result in different classification of the word forms. The members of each cluster are thus assigned the same tag.

Previous tagset reductions have been connected with specific mutual dependence between each two of the tagsets. The tagsets mappings have to preserve the patterns, the (ir)regularities of the language.

[13] The first results of the learning process were obtained by selecting the first morphemic tag from the list of tags.
[14] The major par of this subsection is taken from (Hladka, Ribarov 1998)
[15] The subscripted number denotes the cardinality of the tagset.
[16] BMM - Bigram Markov Model, RB - Rule-based, MMFS - Markov model realized by finite state automata

If the training corpus is annotated with the POS TAG_{1171} we can observe results of the rule-based approach given in Table 6.

Order	Description of the Rule	Success (%)
1	Swap the dependency between ZIP and NFS4A	33.67
2	Swap the dependency between ANS51A and NFP5A	34.53
3	Swap the dependency between PDFS2 and NFS6A	35.14
4	Swap the dependency between PQFIP1 and VPS3A	35.74
5	Swap the dependency between RV7 and VPS3A	36.27
6	Swap the dependency between ANS51A and NIS4A	36.81
7	Swap the dependency between ANP71A and NFP7A	44.56
8	Swap the dependency between PQFMP1 and VPP3N	44.76
9	Swap the dependency between ANS53A and NFS6A	44.96
10	Swap the dependency between ANS61A and NNS6A	45.16
...		
11	Swap the dependency between AFS71A and NMS6A	47.18
12	Swap the dependency between ANS61A and NOMORPH	47.38

Table 6

The situation changes rather dramatically if we train on the reduced tagset POS TAG_{34} as shown in Table 7.

Order	Description of the Rule	Success (%)
1	Swap the dependency between ADJ and NOUN	44.38
2	Swap the dependency between CM and CONJ	46.00
3	Swap the dependency between PSE and VERB_PRI	47.48
4	Swap the dependency between ADV and VERB_PRI	48.74
5	Swap the dependency between PROP and VERB_PRI	49.58
6	Swap the dependency between ADJ and NOUN	50.42
7	Swap the dependency between CM ADJ	51.29
8	Delete the dependency between ADJ and CM	52.22
...		
9	Add a dependency between PUNCT and PROP	64.84
10	Delete the dependency between PRON_INS and PREP	65.02

Table 7[17]

[17] The rules in Table 5 and Table 4 are not the complete set of rules.

Not all of the rules result in a sentence structure with a precise grammatical explanation. The „strange“ ones are there to combine with the others in order to correct the structures of the previously applied rules. Although some of the rules are obvious and we believe that a human would derive the same rules, still the grammatical meaning of the rules can be evaluated only when analyzing the result of the application of the whole list of the rules. Let us examine the relation and dependence of the reduced and non-reduced tagset on the selected rules as given in the above tables. One of the obvious rules in Table 7 is rule 1 (and rule 6; the rules might repeat; during their repetition their influence has a different scope depending on their position in the list). To cope with a more distinct situation in a more specific POS TAG_{1171}, the algorithm produces more rules in order to capture the relation between an adjective and a noun: rules 2, 6, 7, 9, 10 and 11 in Table 6.

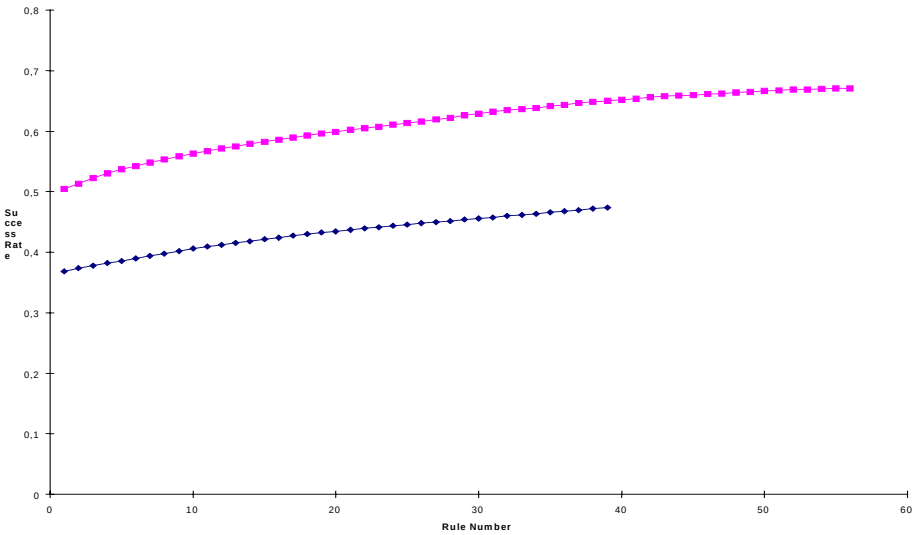

Figure 9: The saturation of the learning process

Undoubtedly, the reduced tagset brings better absolute values. We would like to note that the nodes within the syntactic structure could be returned to the corresponding values from the full tagset. Thus, no information has been lost.

As for the Rule-Based application for syntactic structure extraction, the reduced tagset leads to a much better start on the learning curve (Figure 9). After the saturation of the process of learning, which comes after several tens of rules have been learnt, the learning might continue after one switches to the more expanded tagset, namely the full tagset.

3.3. Syntactically sensitive tag set reduction

Being sensitive to the tagset size, the rule-based approach operates better on smaller tagsets. Therefore if a reduction of a tagset does not distort the syntactic characteristics of the input, one should perform this reduction.

A reasonable reduction of the morphemic tagset for Czech should respect the following facts:

- With nouns, the difference of nominative, accusative, dative, genitive from the other cases should be preserved.
- Adjectives in nominative should be distinguished from the other adjectives.
- Adjective pronouns should be distinguished from nominal ones.
- Adjective numerals should be distinguished from nominal ones.

– Two verbs deserve special considerations. Those verbs are "mít" (to have) and "být" (to be). As for the verb "být" in the sentences where more than one verb occurs, more detailed distinctions should be made, which are:
  - "být" as an auxiliary verb in future tense
  - "být" as an auxiliary verb in past tense
  - "být" as an auxiliary verb in a conditional clause
  - if no other verb form occurs, then "být" is a copula or a verb of existence.
– Within sentences with more than one verb a distinction between coordinating and subordinating conjunctions should be made.
– Prepositions should be divided into various groups according to the cases they bridge, as for example locative - <preposition> - accusative, instrumental - <preposition> - accusative, instrumental - <preposition> - genitive, etc.
– There is no need of subcategorizing adverbs, interjections and particles, therefore in these cases it is sufficient to yeald only the main POS category.

4. Second attempt for parsing rules modifications

There are no directions according to which a rule is designed. Nevertheless there are several guidelines the rules follow:

- The form of the rule is limited by the needs of the operations to be performed, thus by the aim to be achieved.
- The rule should be computationally achievable within the time and space determined by the application needs.
- The rule is suited to the formal framework of the language theory. It is the design of the templates of the rules which are language dependent although the RBA as such is language independent.

The first set of rules applied for Czech (Subsection 2.2) followed the logic of the rules for English. They performed, conditioned under various triggering environments, only two simple actions: addition or deletion of constituency boundaries resulting in their left or right shift, see Figure 10.

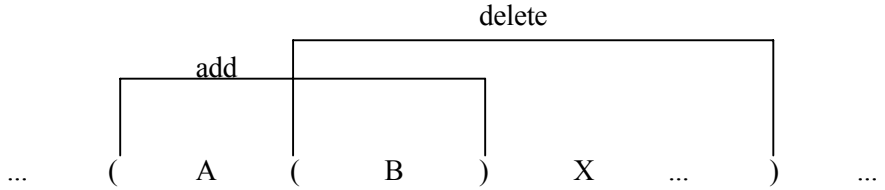

Figure 10

As shown in the previous sections, such rules do not guarantee a significantly high success rate. A new exhaustive set of rules was designed. They were designed in such a way that they cover the basic operations that can be performed on a dependency tree. The interpretation of the rules was modified so that they operate on the tree representation of the sentence: there is a left paren between A and B also in

the cases as in A (C, B), since the left paren is understood as a dependency. Thus, the changes the rule performs are always +/- one tree level[18].

4.1. Modified rule templates for parsing of Czech - second attempt

The second attempt to modify the rule templates is a result of a broad study and is summarized in Table 8.

Rule	**Explanation**
ADD (LEFT A	Make A to be the governing node of the sister nodes of A
ADD (RIGHT A	Push A one level lower in the dependency path
ADD (BETWEEN A B	If A and B are sister nodes, apply ADD (RIGHT A.
ADD) LEFT A	Move A one level up in the dependency tree.
ADD) RIGHT A	The left sister node of A becomes its governing node.
ADD) BETWEEN A B	If A and B are sister nodes, apply ADD) RIGHT A
DEL (LEFT A	A becomes a sister node of what has been its governing node.
DEL (RIGHT A	The subtree of A is moved one level up in the tree.
DEL (BETWEEN A B	If there is a dependency between A and B, apply DEL (RIGHT A.
DEL) LEFT A	The right bracket to the left from A and its counter bracket are deleted.
DEL) RIGHT A	The right bracket to the right from A and its counter bracket are deleted.
DEL) BETWEEN A B	If B is the sister node of the governing node of A, apply DEL) RIGHT A.
SWAP (RIGHT A	If the left bracket exists to the right from A, do: DEL (RIGHT A and ADD (LEFT A.
SWAP (LEFT A	If the left bracket exists to the left from A, do: DEL (LEFT A and ADD (RIGHT A.
SWAP) RIGHT A	If the right bracket exists to the right from A, do: DEL) RIGHT A and ADD) LEFT A.
SWAP) LEFT A	If the right bracket exists to the left from A, do: DEL) LEFT A and ADD) RIGHT A.
SWAP (BETWEEN A B	If there is a dependency relation between A and B, change its direction from A to B to B to A.
SWAP) BETWEEN A B	If there no dependency relation between A and B and B is one level higher than A, swap A and B.
SWAP COMMA BETWEEN A B	If A and B are on the same tree level, exchange the subtrees they govern between them.

[18] The complexity of the process of learning would significantly increase if rules operating on nodes not being directly dependent are used.

MAKE ROOT LEFT A	Make node A to be the root of the tree.
MAKE ROOTFIRST BETWEEN A B	If A and B are sister nodes, A becomes a root node.
MAKE ROOTSECOND BETWEEN A B	If A and B are sister nodes, B becomes a root node.
MAKE GROUP LEFT A	All nodes governed by A become immediate sister nodes of that node (the structure collapses so that everything is governed by A).
MAKE GROUPFIRST BETWEEN A B	If A and B are sister nods, all nodes governed by A become immediate sister nodes of A.
MAKE GROUPSECOND BETWEEN A B	If A and B are sister nodes, all nodes governed by B become immediate sisters of B.
MAKE DEPENDENCE BETWEEN A B	If there is no dependency relation between A and B, establish it. A and B can be anywhere in the tree (exception to the 1 level distance application of the rules).

Table 8: Modified rule templates

Some general remarks on the rules from Table 8:

- all of the rules follow the form as presented earlier: `<action> <what> <where>`.
- LEFT means 'to the first suitable position to the left' in the linear form of the tree.
- RIGHT means 'to the first suitable position to the right' in the linear form of the tree.
- Each sentence is represented in its linear form. The rules operate on the linear form but their interpretation is as if they had operated on a tree.
- As for the transformation, A and B represent the whole subtrees they govern.
- There are no limitations whether the rules can or cannot create non-projective syntactic trees, nevertheless the rules are not designed to identify them and treat them in a different manner.

4.2. Example of an application of the new rules

In order to illustrate an application of the main types of the rules, let us consider the linearized tree structure A (B(C, D), E(F, G), H, I, J(K (L, M))) (Figure 11)

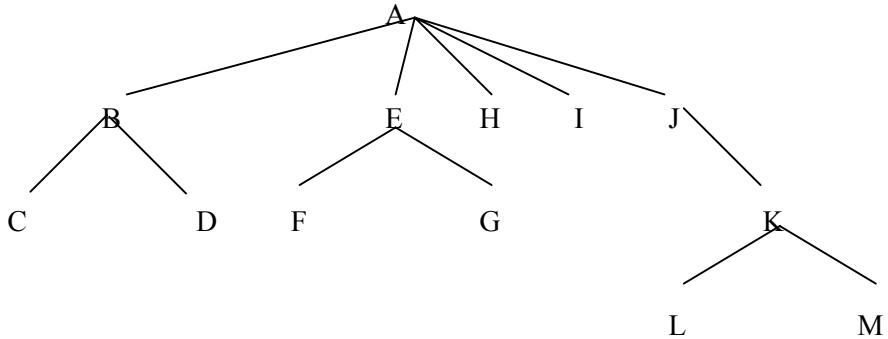

Figure 11

If the rule ADD) LEFT D is applied, the resulting structure is

A (B(C), D, E(F, G), H, I, J(K (L, M))) (Figure 12)

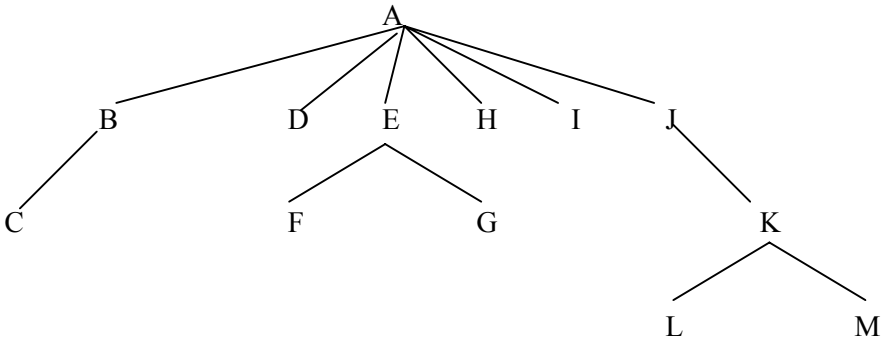

Figure 12

SWAP COMMA BETWEEN G K changing the original tree into

A (B(C, D), E(F, K (L, M)), H, I, J(G)) (Figure 13)

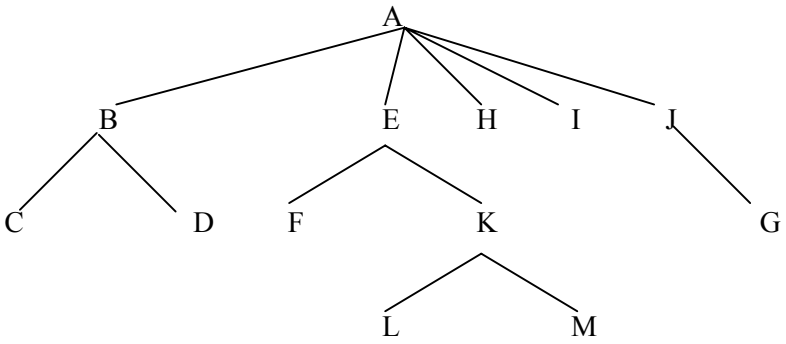

Figure 13

The rule SWAP (BETWEEN J K makes the following changes:

A (B(C, D), E(F, G), H, I, K(J, L, M)) (Figure 14)

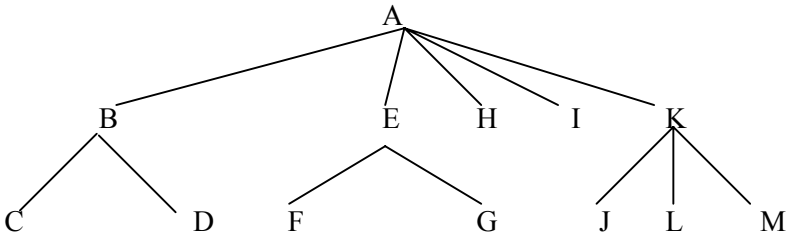

Figure 14

Finally, 'ADD) LEFT I' adds the right bracket which means moving all of the sister nodes to the left from node I one level deeper within the dependency structure :

A (B(C, D), E(F, G, H) I, J(K (L, M))) (Figure 15)

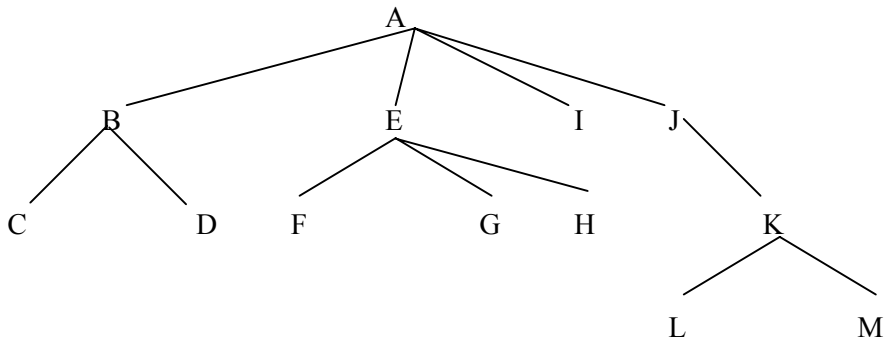

Figure 15

The performance of the remaining set of rules is not presented here since the changes they cause on the tree structure are deductible from what has been presented in Figures 11 to 15.

The last seven rules from Table 8 did not improve significantly the success rate while parsing and therefore they are not exemplified here. One could easily imagine the tree transformations they perform.

4.3. Best rules

The list of the best rules learned from 100 sentences is presented in Table 9. The tagset has been reduced and only the first two characters (positions, POS and sub POS) are taken into consideration. The success rate presented in Table 9 is the cumulative success rate of the rule during the process of learning. Experiments with more than 100 sentences (200 and 300 sentences) were also carried out. None of those experiments resulted in a list of more successful best rules than the one presented here.

Rule	Success
SWAP (RIGHT AF	0.415543
SWAP (LEFT JE	0.436162
SWAP (BETWEEN ZI JS	0.448850
SWAP (RIGHT PS	0.458366
SWAP (RIGHT PQ	0.467883
SWAP (RIGHT DB	0.477399
SWAP (RIGHT PD	0.486122
SWAP (RIGHT DG	0.493259
SWAP (BETWEEN ZI VP	0.500396
DEL (RIGHT AF	0.505155
SWAP (RIGHT PR	0.509913
SWAP (RIGHT VC	0.515464
ADD (BETWEEN R4 NF	0.519429
SWAP (BETWEEN ZI JE	0.523394
SWAP (BETWEEN ZN JS	0.527359
SWAP (BETWEEN PP VP	0.530531
SWAP (RIGHT NO	0.533703
ADD (RIGHT R2	0.536875
SWAP (RIGHT AM	0.540048
DEL (RIGHT RV	0.542427
DEL (BETWEEN ZI NF	0.544806
SWAP (BETWEEN NI DB	0.547978
DEL (RIGHT T|	0.550357
DEL (BETWEEN DB DB	0.552736
SWAP (BETWEEN JE VV	0.555115
SWAP (RIGHT AI	0.557494
SWAP (BETWEEN VF VR	0.559873
ADD (BETWEEN PD DB	0.561459
SWAP (BETWEEN JS NM	0.563045
DEL) RIGHT ZI	0.564631
ADD (BETWEEN RV NI	0.566217
DEL (BETWEEN JE R7	0.567803
SWAP COMMA BETWEEN VP NF	0.569389
SWAP (RIGHT PE	0.570975
SWAP (BETWEEN NF VR	0.572562
SWAP (BETWEEN VM NO	0.574147
SWAP (BETWEEN VM VR	0.575734
ADD (BETWEEN JS VP	0.577320
DEL) BETWEEN VR ZI	0.578906
DEL (RIGHT DG	0.580492
SWAP (BETWEEN JE JS	0.582078
ADD (BETWEEN VP NI	0.583664
DEL (BETWEEN ZI NM	0.585250
SWAP (BETWEEN JE ZS	0.586836
SWAP (BETWEEN T VP	0.588422
ADD (BETWEEN JE AF	0.589215
ADD) BETWEEN DB R4	0.590008
SWAP (BETWEEN ZI VG	0.590801
ADD (BETWEEN JE NF	0.591594
ADD) BETWEEN AF DB	0.592387
SWAP (BETWEEN PD DG	0.593180
SWAP (BETWEEN DG VP	0.593973
ADD (BETWEEN VP NM	0.594766
DEL) LEFT NM	0.595559
SWAP (BETWEEN JE RV	0.596352
SWAP (BETWEEN JS PD	0.597145
ADD (BETWEEN ZI NO	0.597938
DEL (RIGHT PA	0.598731
ADD (BETWEEN NO ZI	0.599524

ADD (BETWEEN VM NF	0.600317
ADD (BETWEEN ZN AB	0.601110
SWAP (BETWEEN DB AV	0.601903
SWAP (BETWEEN VP VS	0.602696
SWAP (BETWEEN AF NO	0.603489
ADD (BETWEEN NO AF	0.605075
SWAP) BETWEEN AF NF	0.606661
SWAP COMMA BETWEEN ZI PI	0.607454
DEL (RIGHT PL	0.608247
SWAP (BETWEEN NF VR	0.610626
SWAP COMMA BETWEEN NI NI	0.611419
ADD (BETWEEN NI DB	0.612213
ADD (BETWEEN NI R4	0.613006
ADD (BETWEEN NM AB	0.613799
SWAP (BETWEEN NI RV	0.614592
SWAP (BETWEEN R2 RV	0.616178
DEL (BETWEEN RV R3	0.616971
ADD (BETWEEN R4 NN	0.617764
SWAP (BETWEEN R4 R4	0.618557
ADD (BETWEEN DG PD	0.619350
SWAP (BETWEEN CG NF	0.620143
DEL (BETWEEN JE PD	0.620936
ADD (BETWEEN JE AF	0.622522
ADD) BETWEEN PQ NF	0.623315

SWAP (RIGHT VU	0.624108
DEL (RIGHT PN	0.624901
ADD) BETWEEN VP NF	0.625694
SWAP (BETWEEN PQ JS	0.626487
ADD) BETWEEN AV NM	0.627280
DEL (RIGHT AC	0.628073
ADD (BETWEEN R3 NF	0.628866
ADD (BETWEEN VF R4	0.629659
DEL (BETWEEN VF AF	0.630452
ADD (BETWEEN VF JE	0.631245
DEL (BETWEEN NI R3	0.632038
DEL (BETWEEN NM NM	0.632831
SWAP (BETWEEN R2 ZI	0.636003
ADD (BETWEEN VP NM	0.636796
SWAP (BETWEEN ZI CX	0.637589
SWAP (BETWEEN JE VS	0.638382
SWAP (BETWEEN DB JE	0.639968
SWAP (BETWEEN VF VS	0.641554
ADD (BETWEEN R2 NN	0.642347
DEL (BETWEEN ZI RV	0.643140
SWAP (BETWEEN VP VR	0.643933
DEL (BETWEEN NI AB	0.644726

Table 9: List of best rules obtained from the learning process on 100 sentences

4.4. The error function as it is and the error function as it could be

The error function is the key determiner of how good a certain structure is when compared to another one. The error function used in all of the experiments with syntactic dependency structures of Czech (not only for RBA) in all of the works presented so far determines the ratio between the correct and the total number of dependencies. Such an error function will make no difference between tree structures from Figure 16 in terms of the values of their nodes.

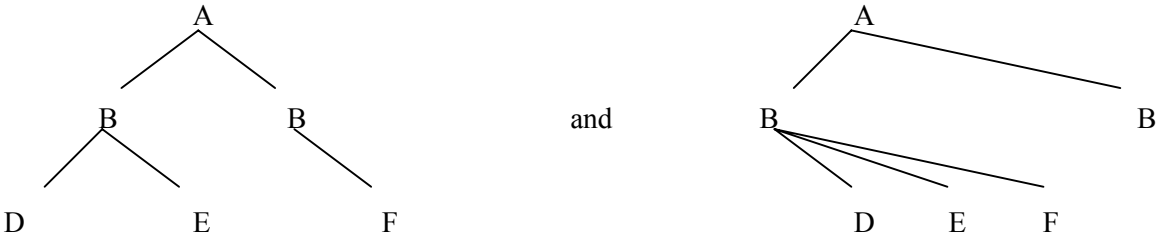

Figure 16

Secondly, there are situations where this error function is "the bad guide" for the process of learning. If a dependency relation A - B (Figure 17 (a)) has to be compared to the dependencies A - C and C - B of the tree path A - C - B on the one hand (Figure 17 (b)), and on the other hand to the dependencies C - A and C - B of the subtree C (A, B) (Figure 17 (c)), obviously the success rate will yield 0% in both of the cases. But, it seems that there is a dependency (non-direct) relation between A and B in the first case, and that one should be able to distinguish between the first and the second case.

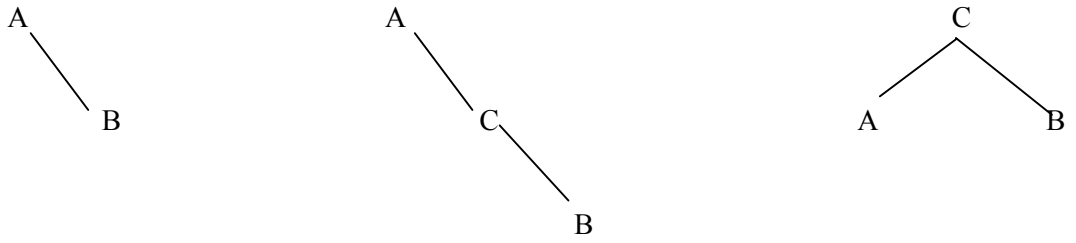

Figure 17

For a transformation to be performed by the stated rules it is "easier to make" B directly dependent on A at Figure 17 (b) since B is in the subtree of A, while it is more difficult to achieve the required dependency in the case of Figure 17 (c) since B and A are two different subtrees which are not necessarily next to each other. Also in Figure 17 (b) it can be considered that B is not directly dependent but remains to be at least distantly dependent on A.

Thirdly, it is neither directly nor approximately possible to compare the success of the algorithms used for dependencies to the one used for constituency for an application of the mentioned error function. Assuming that a constituency structure can be derived from a dependency tree and a predefined hierarchy of the non-terminals, then the derivation history of the constituency is also determined. Each terminal node is specified by the path it determines from the root of the tree. Though not linguistically similar, paths can be enumerated also from the dependency tree. Thus, comparing paths instead of dependencies can make the absolute success rates "more comparable".

Therefore, I propose to count the error rate in a more diversified manner than only in the classical way. The proposed approach consists in including the length of the dependency (the direct dependency could have the value of 1, while the non-direct dependency could be given a success rate as a decreasing function of the distance). Two functions are directly offering themselves to fulfil this aim: reciprocal distance $f(n) = \frac{1}{n}$ and negative exponential distance $g(n) = \frac{1}{e^{n-1}}$. The latter "punishes" the non-direct dependency distance more than f(n) does.

For the previous comparison example, A - B to A- C - B, the direct dependency A - B exists as a non-direct dependency in A - C - B; a "weight" or "success" of $g(n=2) = \frac{1}{e} \cong 0.37$, opposed to 1 if A - B had to be compared to A - B.

In order to be able to calculate the success rate a V(T, F), the *value* of a tree T, should be defined, which would allow for the calculation of the ratio of existing over possible values (precision). Let $V(T,F) = \sum_{\substack{\forall path \\ of T}} F(path)$, where F(n) is either f(n) or g(n).

If T is a tree describing a correct syntactic structure then its value is calculated as V(T, F); it will always be higher than any incorrect variant.

Let us compare the structure T_1: A - B - C - D to the structure T_2: A(B (D), C). Let the latter be the true structure. Its value $V(T_2, g) = 3 + 0.37 = 3.37$. If T_1 is to be compared to T_2 one may compare any of the possible paths in T_1 (A - B, B - C, C - D, A - B - C, B - C - D, A - B - C - D) to all of the paths of T_2 (A - B, B - D, A - C, A - B - D). The set of comparable paths is $T_{1,2}$(A - B, A - B - C, B - C - D) with its value $V(T_{1,2}, g) = 1 + 0.37 + 0.37 = 1.74$. The success rate is then $V(T_{1,2}, g)/ V(T_2, g) = 1.74/3.37 = 51.63$ %.

If the rule-based error driven learning includes this error function, the rules presented in Table 10 can be achieved (as in the previous case the learning process was on 100 sentences; the same set of sentences).

Rule	Success
SWAP (RIGHT AF	0.590218
SWAP (RIGHT ZI	0.613243
SWAP (LEFT JE	0.630062
SWAP (BETWEEN PQ VP	0.648547
SWAP (LEFT VR	0.658814
SWAP (RIGHT DB	0.668000
SWAP (RIGHT DG	0.676885
SWAP (RIGHT PS	0.684720
SWAP (RIGHT PD	0.691009
SWAP (RIGHT PR	0.697009
SWAP (BETWEEN ZI DB	0.701609
DEL (RIGHT PP	0.705621
DEL (BETWEEN NI ZI	0.709475
SWAP (BETWEEN NI JE	0.713570
DEL (BETWEEN RV JS	0.716804
DEL (BETWEEN RV R2	0.719927
SWAP (BETWEEN RV VP	0.722952
DEL (RIGHT PQ	0.725858
SWAP (BETWEEN NI ZI	0.728761

Table 10: A fragment of the list of best rules obtained from the learning process on 100 sentences with negative exponential error function

The success rate of the learning process is higher since the way it has been calculated results in higher percentage rates. This higher success rate does not necessarily mean a less errorneous output.

5. Conclusions

Mentioning comparison of outputs, it has been taken into consideration that the constituency based output for English and the dependency based output of Czech are not directly comparable - nevertheless the relative improvement of the initial parsing structure of Czech is not as good as the one for English. Thus, after numerous experiments for Czech it were these experiments that diverted our opinions into the direction that the Rule-based Parsing fails to be a promising algorithm for parsing of Czech. Nevertheless, the Rule-based Approach as a learning paradigm is not under question here - but the generality contained within the so-called language independence needs revision. As the presented experiments clearly show, it is not enough only analogously to modify the rule templates and to run the "language independent" algorithm.

Acknowledgements

This research is supported by the project number LN00A063 - Ministry of Education of the Czech Republic, the Center for Computational Linguistics, Faculty of Mathematics and Physics, Charles University.

References

Böhmová A. (2001). Automatic procedures in tectogrammatical tagging. *The Prague Bulletin of Mathematical Linguistics 76*, Charles University Press, Prague, Czech Republic.

Brill E. (1993 a). *A Corpus-Based Approach to Language Learning*, A dissertation in the Department of Computer and Information Science, University of Pennsylvania.

Brill E. (1993 b). Automatic grammar induction and parsing free text: A transformation-based approach, In *Proceedings of the 31st Meeting of ACL*, Columbus, Ohio.

Brill E. (1992). A simple rule-based part of speech tagger. *In Proceedings of the 3rd ANLP Conference*, ACL, Trento, Italy.

Collins M., Hajič J., Brill E., Ramshaw L., Tillmann C. (1999). A Statistical Parser of Czech. In *Proceedings of 37th ACL'99*, pp. 505-512, University of Maryland, College Park, June 22-25.

Elworthy D. (1995). Tagset design and inflected languages. In *Proceedings of the ACL Sigdat Workshop*, pp. 1-9, Dublin, Ireland.

Hajič J., Ribarov K. (1997). Rule-based Dependencies. In *Proceedings of the Workshop on the Empirical Learning of Natural Language Processing Tasks*, pp. 125-136, Prague, Czech Republic.

Hladká B., Ribarov K. (1998). Part of speech tags for automatic tagging and syntactic structures, In *Issues of Valency and Meaning - Studies in honour of Jarmila Panevova*, Karolinum, Charles University Press, pp. 226-237, Prague, Czech Republic.

Ribarov K. (1996). *Automatic Natural Language Grammar*. MSc. Thesis (in Czech), Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic.

Sgall P., Hajičová E., Panevová J. (1986*). The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Reidel Publishing Company, Dordrecht, Netherlands, Academia, Prague, Czech Republic.

The Prague Dependency Treebank ver. 1.0 CD-ROM (2001). Issued via the Linguistic Data Consortium, Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic.