The Faculty of Mathematics and Physics
Charles University

Problems of Robust Parsing of Czech

Vladislav Kuboň

PhD Thesis

Institute of Formal and Applied Linguistics
Praha, 2001

Disertační práce byla vypracována v rámci doktorandského studia, které uchazeč absolvoval v Ústavu formální a aplikované lingvistiky Matematicko-fyzikální fakulty Univerzity Karlovy v Praze v letech 1996-2001.

Uchazeč:	RNDr. Vladislav Kuboň
Školitel:	Prof. PhDr. J. Panevová, DrSc. Ústav formální a aplikované lingvistiky, MFF UK
Školicí pracoviště:	Ústav formální a aplikované lingvistiky, MFF UK Malostranské náměstí 25, 118 00 Praha 1
Oponenti:	Prof. Patrice Pognan CERTAL INALCO, Paris
	Doc. RNDr. Vladimír Petkevič, CSc. Ústav teoretické a komputační lingvistiky, FF UK

Contents

Introduction

It is not surprising that the problem of syntactic parsing of natural languages belongs to the most popular fields of computational linguistics. An automatic syntactic parser is a key element of a wide range of natural language processing systems. The quality of various types of research projects directly depends on the quality of parsing modules contained in these systems.

Natural language parsing is also a very interesting research field. On the one hand, it is sufficiently close to the problem of parsing formal languages, which has been studied by computer scientists for a very long time and which has therefore been investigated to a great depth (at least for context-free languages). On the other hand, the syntactic analysis of natural languages does not allow for a direct application of the results of the theory of formal languages and thus provides more than a sufficient challenge for those wishing to develop new methods and approaches both in theory and in implementation.

The practical experience in the field of natural language parsing shows that every project aiming at a (as much as possible) complete coverage of the natural language under consideration has to cope with imperfections of the input. Even in case the input texts were thoroughly checked (and grammatically corrected) by human reviewers (for example in journals or newspapers), it is often possible to encounter a grammatical error. The investigations carried out during the Joint Research Project PECO 2824 have shown that printed texts contain numerous errors of a wide range of error types - cf. [Petkevič 95]. Even worse is the situation with spoken language. It is therefore quite clear that if we aim at parsing any input sentence from a real text in the given language, we have to build a robust parser.

This work approaches the problem of building a robust parser for Czech (as a typical language with high degree of word-order freedom) from two points of view. The first one is the theoretical point of view, in which we are aiming at a more exact specification of the term robust parsing, at its connection to related fields (natural language analysis, grammar checking etc.) and at the parsing complexity of approaches designed here. The second point of view is that of an implementation of a sample metagrammar covering (a subset of) syntactic rules for Czech. The metagrammar represents in fact the first implementation of a rule-based robust parser for Czech. It serves mainly for two purposes – the first one is the purpose of demonstrating the usability of an approach described in the theoretical part, the second one is to provide stimuli for the subsequent development of even better and more adequate theoretical description of the problem.

For the sake of consistency of the text the dissertation contains not only the description of the work done solely by the author, but also the results of a collaborative work carried out during the past six years by the research team of which the author was one of the members. However, the parts that are a result of a cooperative work are explicitly marked as such.

The first chapter introduces the basic notions, which are necessary for the description of the aims and goals of this thesis.

The second chapter aims at the history of the automatic parsing of Czech. It stresses especially the systems, which may be considered a direct predecessors of our system.

Basic features of our approach are described in the third chapter.

The fourth chapter describes individual components of the system of a robust parser of Czech.

Second part of the description of the basic notions can be found in the fifth chapter. It provides additional definitions allowing to formulate important conclusions concerning complexity estimations in the following chapter.

The seventh chapter contains a description of the software environment used for the implementation of the system.

The format of the syntactic dictionary is described in detail in the eighth chapter.

The ninth chapter contains a thorough description of one of the key elements of the system – the metagrammar. It also contains the description of the linguistic phenomena the system is (or is not) able to handle.

Important facts about the process of testing and debugging the metagrammar are to be found in the tenth chapter.

The eleventh chapter contains a thorough description of the results of parsing a set of sentences used as a testbed in the process of the development of the metagrammar.

The last, the twelfth one, chapter is devoted to a description of a method for analysis of the structure of clauses in complex sentences. This method was developed as an attempt to solve some of the problems encountered in the process of the system development and discussed in the eleventh chapter.

The definitions from the third and fifth chapter were developed as a result of a joint effort with Martin Plátek, Tomáš Holan and Karel Oliva and had already been published in several papers. The software environment described in the seventh chapter was developed by Tomáš Holan. The format of the syntactic dictionary was also to a large extent influenced by the work of H. Skoumalová done in the project identified as JEP PECO 2824 Language Technologies for Slavic Languages.

Chapter 1: Basic notions

In this chapter we would like to describe the basic notions used throughout the whole text. First of all, let us define the basic data structure we are going to use both for visualization of the results of the parsing process and also for theoretical purposes. This data structure is a traditional means for the visualization of the syntactic structure of a sentence in the European continental linguistic dependency oriented syntactic tradition. A number of different formalizations exist, e.g. [Nebeský 72], [Kunze 72]. Our definition contains more information than the traditional ones and in this manner better reflects the needs of subsequent chapters. It was originally defined in [Holan et al. 2000].

Definition 1.1 (Dependency-tree, D-tree)
We define the tree T to be a dependency tree (D-tree) over the sentence w = a1...an, if T contains exactly n nodes, and for each i ∈ {1..n} there is a node of the tree T in the form [ai,i,vi,di], where we denote the individual parts ai,i,vi,di of the node as follows:

a_i is the *symbol* of the node (the word label),

i is the *horizontal index* (the surface position, counted from left),

v_i is the *vertical index* (the distance, measured by the number of edges, of the given node from the root),

d_i is the *dominance index* (the horizontal index of the governor),

and the following holds:

- $v_i,d_i \in \{0..n\}$; $v_i=0$ if and only if $d_i=0$, in which case the node $[a_i,i,0,0]$ is the root node of the tree *T*.
- if $u=[a_i,i,v_i,d_i]$ and $d_i \neq 0$, then there exists exactly one node $v=[a_{d_i},d_i,v_i-1,d_{d_i}]$. The (ordered) pair *(u,v)* creates the (single, oriented) edge of the tree *T*, incident to the node *u*.

Let us illustrate the definition using the following example.

Example 1.1
Nepodařilo se otevřít zadaný soubor.
[Lit.: It_was_not_possible Refl. to_open specified file.]
(It was not possible to open the specified file.)

There are the following nodes in the D-tree of this sentence:
u_1=[Nepodařilo,1,0,0], u_2=[se,2,1,1], u_3=[otevřít,3,1,1], u_4=[zadaný, 4,3,5], u_5=[soubor,5,2,3]

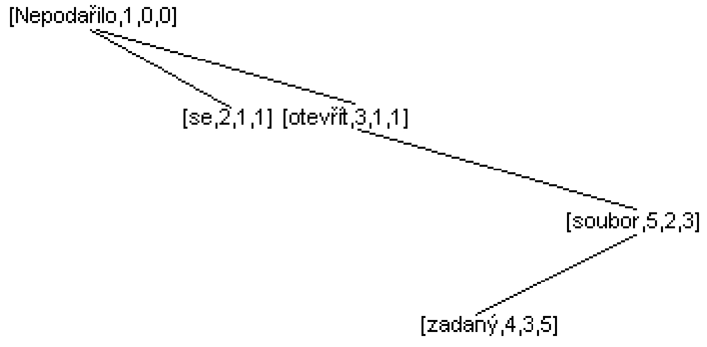

Fig. 1.1. D-tree T_1 for the sentence from example 1.1

In the following text we are also going to speak very often about a very important property of dependency trees, namely the nonprojectivity. Let us first define the term of *projection of a node of the D-tree* (in some of our previous papers we have used the term *coverage*), which then allows us to formulate the definition of the *projectivity/nonprojectivity* term in a manner suitable for further definitions of *measures of nonprojectivity*.

Definition 1.2 (Projection of a node of a D-tree)
Let *T* be a D-tree and let *u* be a node of *T*. The set of horizontal indices of all nodes *v* of the tree *T* such that there exists an (oriented) path from *v* to *u* will be called the projection of *u* within *T* and marked off as *Cov(u,T)*. When defining *Cov(u,T)*, we take into consideration also the empty path, hence *Cov(u,T)* always contains the horizontal index of *u*.

Projections of individual nodes of the D-tree T_1 from the Fig.1.1 look as follows:
$Cov(u_1,T_1)=\{1,2,3,4,5\}$, $Cov(u_2,T_1)=\{2\}$, $Cov(u_3,T_1)=\{3,4,5\}$, $Cov(u_4,T_1)=\{4\}$, $Cov(u_5,T_1)=\{4,5\}$

The next necessary step is the definition of a *hole* in the projection of a node.

Definition 1.3 (Hole (in the projection of a node))
Let *T* be a D-tree over the sentence $w = a_1...a_n$, let *u* be a node of the tree *T* and let

$$Cov(u,T)= \{i_1,i_2,...,i_k\},\ i_1 < i_2 < ... < i_{k-1}<i_k.$$

For $1 \leq j < k$ we say that the pair (i_j,i_{j+1}) creates a *hole in Cov(u,T)*, iff $i_{j+1} - i_j > 1$.

Let us now demonstrate the notion of a hole using only a slightly modified sentence from the previous example:

Example 1.2
Zadaný soubor se nepodařilo otevřít.
[Lit.: Specified file Refl. it_was_not_possible to_open.]
(The specified file failed to open.)

There are the following nodes in the D-tree of this sentence:
u_1=[Zadaný,1,3,2], u_2=[soubor,2,2,5], u_3=[se,3,1,4], u_4=[nepodařilo, 4,0,0], u_5=[otevřít,5,1,4]

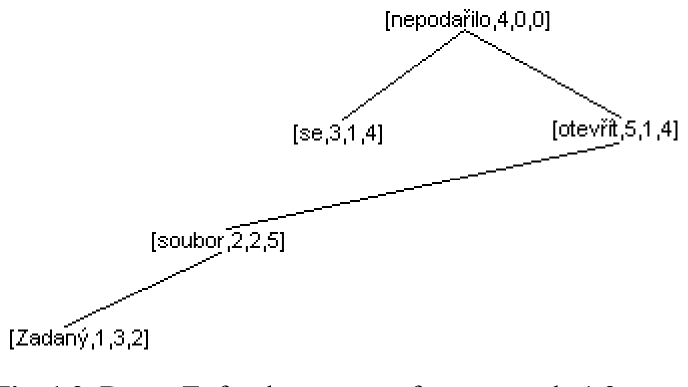

Fig. 1.2. D-tree T_2 for the sentence from example 1.2
The projections of individual nodes of the D-tree T_2 from the Fig.1.2 look as follows:
$Cov(u_1,T_2)=\{1\}$, $Cov(u_2,T_2)=\{1,2\}$, $Cov(u_3,T_2)=\{3\}$, $Cov(u_4,T_2)=\{1,2,3,4,5\}$, $Cov(u_5,T_2)=\{1,2,5\}$
The projection of the fifth node $Cov(u_5,T_2)=\{1,2,5\}$contains a hole.

The last step we need to make before we are able to define the terms of projectivity and nonprojectivity is the definition of the following measure:

Definition 1.4 (Measure dNh)
Let *T* be a D-tree over the sentence *w* and let *u* be a node of the tree *T*. We define the measure *dNh(u,T)* (*number of holes in a dependency subtree rooted in u*) as the number of holes in *Cov(u,T)*.

It is now easy to formalize the traditional terms of projectivity and nonprojectivity of a dependency tree (for previous usage of these terms cf. [Marcus 65], [Kunze 72], [Nebeský 72]) using the notions defined in previous definitions.

Definition 1.5 (Projectivity, Non-projectivity)
Let *T* be a D-tree over the sentence *w*. The number *dNh(T)*, equal to the maximum of $\{dNh(u,T);$ $u \in T\}$, will be called *the magnitude of non-projectivity* of the sentence *w* with the structure *T*.
If *dNh(T)=0* holds, we say that the sentence *w* with the structure *T* is *projective*, otherwise we say that the sentence *w* with the structure *T* is *nonprojective*.

According to our definitions the sentence from the Example 1.1 (*Nepodařilo se otevřít zadaný soubor*) is projective and the sentence from the Example 1.2 (*Zadaný soubor se nepodařilo otevřít*) is nonprojective.

The next very important set of terms used in the subsequent text is connected with our ultimate goal, namely with the robust parsing of Czech. It is often the case that different people in different circumstances understand the term of natural language analysis or natural language parsing in various ways. For example, for statisticians the term (stochastic) natural language parser means a device that always provides just one syntactic structure as its result for any input sentence. For people working with theories of formal parsing, the result a parser should provide is a set of structures, each of which represents a plausible analysis of the input according to a certain grammar. On the other hand, for linguists the result of (syntactic) parsing should be a set of structures representing all plausible readings of the input sentence. The difference between the last two variants consists in that linguists very often exploit not only syntactic knowledge, but take also into account the semantics, pragmatics or even the real world knowledge when deciding about the acceptability of resulting structures. On the contrary, from the point of view of "formalists" the only criterion of acceptability of parsing results is the grammar according to which the input is parsed. Linguists are typically able to assign various degrees of acceptability to various structures representing the results of natural language analysis, they even may be able to give an answer to the question what's the "best" result out of the set obtained as a result of the parsing. From the point of view of a parsing theory, there is nothing like a "better" or a "worse" result, all structures obtained as a result of parsing are equally acceptable.

In order to overcome this terminological uncertainty we have decided to stick to the following terminology:

We are going to use the term *natural language analysis* for a complex process involving not only syntactic analysis, but also semantics, pragmatics and real world knowledge. The result of *natural language analysis* is typically one (the best, most acceptable) sentence structure. A typical example of *natural language analysis* is the tectogrammatical level of the Prague Dependency Treebank [Hajičová et al. 99], the result of human–made analysis taking into account all possible sources of knowledge and information (including very broad context in which a particular sentence appeared).

The term *(syntactic) parsing* will denote the process of analysis according to a grammar (or metagrammar) containing (syntactic) rules and constraints. If the rules and constraints capture the rules and constraints of the surface syntax, we talk about a *surface (syntactic) parsing*. The result of *(surface) parsing* is a set of structures describing all possible syntactically acceptable variants of the analysis of a particular input sentence. None of the resulting structures is considered to take precedence or to be "better" or "worse" than the rest.

When we are concerned only with whether a particular input sentence belongs to the set of all well-formed sentences of a given natural language, we talk about a *(syntactic) recognition*. This notion is very important e.g. for grammar checking, where we are mainly concerned with the task of distinguishing the well-formed sentences from ill-formed ones. To be sure that the sentence is syntactically correct we need to find just one plausible structure for the sentence. Only in case the sentence is ill-formed we need its complete syntactic analysis in order to be able to locate and classify the syntactic error(s) it contains.

Probably the most important term used in this dissertation is the term *robust (syntactic) parser*. Generally the robustness is understood as a property of a particular system to cope with all kinds of input regardless whether it is ill-formed or well-formed. A typical example of a robust parser in this sense is in fact any stochastic parser. The stochastic analysis does not provide any means for distinguishing ill-formed sentences from well-formed ones, dealing with both types in a uniform way and assigning just one, the most probable, structure to each input sentence. In this way it is even possible to obtain a structure for a sequence of word forms, which is so highly ill-formed that even the native speakers are neither able to assign a structure to this sequence, nor to understand what the sequence of words actually means.

It is quite clear that such a broad robustness may be useful for practical applications, but it has nothing to do with the aim at a more adequate description of a particular natural language. The robustness we are aiming at is different. Our motivation is to simulate how a native speaker of a particular language will be able to cope with ill-formed sentences. If the sentence is so much corrupted that it is both syntactically ill-formed and unintelligible even for a native speaker, then the robust parser also should be unable to parse it. The robustness, as we understand it, has certain limits.

The first goal we aim at is the ability of the robust parser to guarantee syntactic acceptability. While a syntactic parser draws a strict line between sentences belonging to the set of syntactically well-formed sentences of a given natural language and those which are not, a robust parser should be able to draw a similar line between ill-formed sentences which may be corrected according to the syntax of a particular language and those which are definitely syntactically unacceptable.

The second goal of our approach is more general – we aim at creating a theoretical framework allowing to shift this borderline in a consistent and adequate manner by means of the application of a set of general constraints expressed through measures defined in this thesis (and in the papers about our approach published in the previous years). This goal in fact means that in general sense our theory allows to create a scale of parsers with different degrees of robustness or word-order freedom.

One of the main topics of this thesis is the endeavor to describe at least some constraints allowing to achieve these goals. We do not claim that the constraints introduced here are the only constraints suitable for the declared purpose, it is quite clear that several other types of constraints may be formulated in the future. These constraints will then allow to create even a more refined scale of parsers and thus they will also support a more adequate description of syntactic properties of natural languages with a high degree of word order freedom.

At this point it is also necessary to specify the type of constructions which are going to be considered as *syntactically ill-formed* in the following text. The position adopted in this work reflects the fact that very often a sentence is rejected by a human reader for some extrasyntactic reasons. For example it may be unintelligible; it may be stylistically unacceptable; its preferred reading may violate grammatical rules while there is a second, syntactically well-formed reading, with a meaning unacceptable in the given world, etc. Let us demonstrate this fact on a very simple sample sentence:

Koťata chytaly myši. [Kitten(pl.neut.) were_catching(pl.fem.) mice(pl.fem.)]

The problem of this sentence is the collision of syntax with other factors important for human understanding of a sentence, namely with the real world knowledge. At first sight it seems that the sentence is ill-formed due to the wrong (feminine) form of the verb (*chytaly* instead of *chytala*). When we look closer at the reasons for rejection of this sentence, we find out that they are based on the assumption that only *koťata* [kitten] are the acceptable subject of this sentence due to the fact that under normal conditions usually mice are being caught by kitten, not vice versa. The syntactically correct reading suggests the translation “Mice were catching the kittens”, while the human common sense prefers the reading which may be translated as: “Kittens were catching the mice”, even though this reading violates the syntactic rule of subject-predicate agreement. The reason for rejection of this sentence is therefore purely extrasyntactic, it is based on the real life experience.

This example illustrates that it is really necessary to draw a clear borderline between *natural language analysis* and *(surface) syntactic parsing* (as described above). Rather than to attempt at solving a wide spectrum of very complicated problems of *natural language analysis* we would like to concentrate on the question of what can be done if only “pure” syntax is taken into consideration, thus aiming at solving the problem of *syntactic parsing*. The following chapters show that even with such a drastic restriction of scope the problem of *robust parsing* of a language with high degree of word-order freedom is very complicated and complex. Thus, throughout the following text, wherever we are going to refer to ill-formed constructions or sentences, we will have in mind only those constructions or sentences which are ill-formed from the point of view of the syntax only, no other factors are going to be taken into account. This position is similar, for example, to the position taken in the classical literature in [Kunze 75].

Chapter 2: Existing automatic parsers of Czech

In this chapter we are going to describe basic facts about automatic parsers of Czech developed in the past. Two of these systems, namely the parser of Czech developed primarily for the Czech-to-Russian machine translation system and used also in the question-answering system TIBAQ [Hajičová 95] and the parser implemented in the JEP PECO 2824 LATESLAV project, directly influenced our robust parser.

2.1 RUSLAN

The first automatic syntactico-semantic parser of Czech was created for the Czech-to-Russian machine translation project RUSLAN in the late eighties [Oliva 89]. The system was implemented in Colmerauer's Q-systems [Colmeraurer], a chart-parser-like formalism successfully used in the machine translation system TAUM-METEO. Q-systems allow dividing the grammar into modules, where the output of a previous module serves immediately as the input for the following module. Each of the modules consists of a set of rules that in principle describe transformations of tree structures.

The project RUSLAN was conceived of with the aim at the industrial exploitation in translation of handbooks for operating systems of mainframes. The project consisted of about twenty different modules ranging from the morphological analysis of Czech to the morphological synthesis of Russian. The proper syntactic parser of Czech occupied only three modules. It covered a wide range of syntactic phenomena and also exploited some lexical-semantic information. This information was used in semantic constraints of individual grammar rules. These constraints took care of the attachment of “proper” types of dependent lexical items to the governing ones. The following mechanism was used:

1. Each noun had a set of so called “proper semantic features”, marking it as belonging to one of several semantic categories (concrete object, abstract object, human, institution etc.).
2. Some valency frames’ slots contained two other sets of semantic features, the so called “required” and “forbidden” semantic features. The former took care about filling the particular valency slot by a proper type of noun, while the later blocked filling the valency slot by a noun which had among its proper semantic features at least one “forbidden” feature.

The main idea behind the use of semantic features was to decrease the number of trees obtained as a result of parsing. Expressed in the terms defined in the previous chapter, the use of semantic features in fact meant one step from syntactic parsing towards the natural language analysis. The practical experience with semantic features was slightly disappointing. Much more often than reducing the ambiguity of the attachment by filtering out semantically unacceptable candidates the semantic features blocked the attachment of a single unambiguous candidate and thus led to a parsing failure.

The main problem was hidden in the dictionary – with almost ten thousand lexical items in the main dictionary of the system it was very difficult to assign all correct semantic features. A very nice example illustrating the nature of problems encountered is the verb *běhat* [to run]. It contained some required semantic features for its subject – the subject was expected to be a human or a living being. These required semantic features were assigned by language specialists, who did not take into account that the manuals to operating systems are full of programs and operating systems, which also tend to run. It was, of course, very easy to correct the information for a single dictionary item, but in the long run it was more costly to assign and correct semantic features of hundreds of lexical items than simply remove all semantic features and to return to a syntactic parsing not exploiting the lexical semantics.

The problems with semantic features in RUSLAN were also one of the reasons why we have decided to concentrate on “pure” syntax in our robust parser. Our aim is the introduction of a clear invariant serving as a simple criterion of syntactic acceptability. Although the use of “pure” syntax has its drawbacks (for example, it is often very difficult to distinguish between syntax and semantics in a number of language phenomena), it is by far the best candidate for this purpose.

2.1.1 Syntactic analysis in RUSLAN

The first of the three parsing modules covering syntactic analysis of Czech contained mainly rules for parsing simple nominal groups, the only syntactic phenomenon that was considered simple enough to be handled separately from the main module of syntactic analysis.

The second module was the most important one. It contained rules handling the proper syntactic parsing of Czech input sentences. In case this module succeeded, it issued one or more linearized syntactic trees representing the syntactic structure of the sentence. In case for some reason the syntactic parser did not succeed, it passed over to the next module a chain of subtrees covering the whole sentence. In this case the next module containing so called "fail-soft rules" tried to construct a complete syntactic tree from the subtrees created in the second module. There were four most typical reasons for the parsing failure:

1. One or more words were not found in the main dictionary of the system. Even though RUSLAN was designed as a restricted domain system, it was quite often the case that the text contained a word not covered by the dictionaries of the system. If the word was of Greek or Latin origin (technical terminology often has a foreign origin) or if it had a productive ending allowing to guess morphological characteristics of the word, a special module called the transducing dictionary [Bémová, Kuboň 90] took care about it. The idea of a transducing dictionary comes from Z.Kirschner, who used it in the English-to-Czech translation system APAČ, see [Kirschner 88]. If even the transducing dictionary didn’t help, the unrecognized word blocked the parsing of the whole sentence.
2. The grammar contained a series of constraints, which allowed to restrict the nondeterminism of Q-systems and to reduce the number of spurious ambiguities. This needs a bit more detailed explanation. The fact that the interpreter of Q-systems applies the grammar rules in a non-deterministic way (from the point of view of the user it is impossible to determine the order of application of individual grammar rules to the given input) has a direct consequence. It is very often the case that spurious ambiguities are created during the process of application of individual grammar rules. The most typical is the case when the left-hand side and right-hand side modifiers of a particular word are processed without any preference. In that case it is possible to get the resulting dependency syntactic tree by a number of ways (this is, of course, not the case when constituent trees are used, because they describe the derivational history and thus the different order of application of rules results in a different shape of the constituent tree). In order to avoid this situation the grammar used in RUSLAN contained a preference of the left-hand side modifiers at the level of the main (governing) verb of the sentence. The attachment of the right-hand side modifiers was blocked until all left-hand side modifiers were attached. The application of such kind of preference is always slightly dangerous – unsuccessful processing of any of the left hand side modifiers of the main verb blocks the processing of the items on the right hand side of the verb. This was very often a source of parsing failures.
3. The third most common case of the parsing failure was caused by the incompleteness of the grammar or by errors in grammar rules. The process of testing and debugging the grammar that is interpreted in a nondeterministic way is generally very long and complicated. The situation of the parser in the RUSLAN system was even worse due to the fact that it had never been finished, it stopped for the lack of funds shortly after the political changes made the Czech-to-Russian translation useless. Therefore also the grammar of the system was far from complete.
4. Apart from the above mentioned properties of the syntactic parser in the RUSLAN system there was also a certain limit with respect to the type of syntactic phenomena the parser was able to handle. This is connected with the fact that the parser in fact worked in a manner which was similar to what could be described in mathematical terms as a lower estimation of syntactically correct sentences. This statement means that each sentence parsed by a Czech syntactic parser in RUSLAN was guaranteed to be syntactically correct. The system was, of course, not able to handle all syntactically well-formed sentences, so in fact it was able to parse a proper subset of the

language. With corrections of and additions to the grammar during its development this subset grew, but it never covered the entire set of all syntactically well-formed Czech sentences.
It worked in a strictly projective manner, due to the fact that the framework used, Colmerauer's Q-systems, was originally designed for languages with relatively fixed word order and thus it doesn't support a more relaxed word order. The non-projective constructions are much more common in languages with high degree of word order freedom, where different kinds of topicalization or word order variants are much more natural than in natural languages with fixed word order.
The syntactic parser in RUSLAN did not contain any rules attempting to overcome this insufficiency of Q-systems. This is an opposite orientation than should be chosen for a robust parser for languages with a more relaxed order of words. Such a robust parser must aim not only at the analysis of all possible syntactically well formed projective sentences of a given language, it must also be able to handle syntactically ill-formed and/or non-projective sentences. It has to work with relaxed constraints of two types – the constraints on the order of words and the constraints guaranteeing syntactic well-formedness of sentences. In this sense the robust parser aims at the upper estimation of the set of all syntactically well-formed sentences for a given language. Fig. 2.1 shows the mutual relationship of three sets of sentences – those parsable by RUSLAN, by a robust parser and those belonging to the set of all well-formed sentences of a given language.

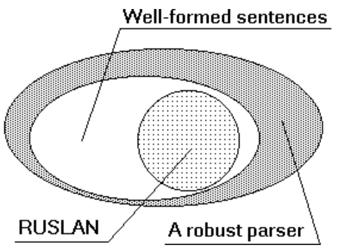

Fig. 2.1. Mutual relationship between the set of all well-formed sentences of a given language and the sets parsed by the RUSLAN system and by the robust parser.

Some of the parsing failures of the module of syntactic analysis in the RUSLAN system (especially those belonging to the first two categories) were handled in the third parsing module. It contained slightly modified rules of the second module. They basically allowed a continuation of parsing the rest of the sentence in case the procedure was blocked on the left-hand side of the main verb. No other technique of robust parsing was used at the syntactic level. The parsing modules did not contain any special rule for processing an ill-formed input.

Even though the RUSLAN system was never completed (there was no commercial demand for Czech to Russian MT system after 1989), its parsing component may be considered as the first automatic syntactic parser of Czech.

2.2 A prototype of a grammar-based grammar checker for Czech

The second important project which had a great influence on our robust parser was the Joint European Project PECO 2824 LATESLAV Language Technology for Slavic Languages. The main task of that project was to develop and implement a grammar-based grammar checker for Czech and Bulgarian. The project started in January 1993 and was finished in June 1996. The theoretical and practical work performed during that project provided a good base for the work on the implementation of a robust parser of Czech. The relevant parts of the project are mentioned in the following chapters of this thesis.

There are several differences between the syntactic parser of Czech in RUSLAN and that in the LATESLAV project. Among the most important ones there is definitely the difference in the ability to handle nonprojective constructions and the degree of attention devoted to the problems connected with parsing ill-formed input. The difference was also in results obtained from both parsers – the parser in RUSLAN was oriented more towards the natural language analysis in the sense defined in the previous chapter, while the grammar checker in LATESLAV worked with syntactic recognition of well-formed sentences and with syntactic parsing of ill-formed ones. On the other hand, both parsers also have many common features, for example they both follow the tradition of Czech linguistics and use dependency trees as a means of representation of sentence structure.

It is possible to say that both projects created certain kind of boundaries for a robust parser of Czech described in this dissertation. The parser used in the RUSLAN system was not robust enough to be widely used for parsing unrestricted text, while the grammar checker used in the LATESLAV system was mainly concerned with the identification and localization of syntactic errors, not so much with providing a complete set of results expected from a syntactic parser in the sense defined in the previous chapter.

2.3 Experiments with a stochastic parser of M. Collins

Both parsers mentioned above are rule-based systems that were created in accordance with the long tradition of Czech theoretical and computational linguistics. However, the attention of researchers shifted towards stochastic systems in the last decade and it concerns the Czech language, too. There were several attempts to create a stochastic syntactic parser of Czech during previous two years. Most of them are not worth mentioning since their success ratio was hardly better than 50%. This might seem to be a good result, until we take into account what this number actually means. It expresses the number of pairs of [governing,dependent] words, which were created correctly with respect to a set of sentences contained in a treebank (hand-parsed by native speakers). This number in fact means that there is a very low probability that at least one sentence in a given set of sentences was parsed correctly.

There is one exception among stochastic parsers of Czech. The best results were achieved during the workshop at the John’s Hopkins University in Baltimore in June of 1998, see [Hajič 98]. The application of the stochastic syntactic parser of M. Collins to the parsing of Czech texts provided much better results, about 80% of correctly assigned pairs. This parser has already proved its quality for English, where the results exceeded 90%. From the point of view of the task of parsing free-word-order languages it is interesting that the stochastic parser of M. Collins achieved about 10% worse results for Czech than for English. This result might indicate that the typological difference between these languages has an influence on the complexity of the task of their parsing. We are far from drawing unjustified conclusions (some people think that the rich inflection of Czech makes the task of parsing easier), but it seems to be clear that parsing a language with such a high degree of word-order freedom as Czech requires slightly different approach both in stochastic and rule-based parsers then the approach used for languages with relatively rigid word order. The following chapters explain in a more detail how we have coped with the challenges of a robust parsing of a free-word-order language.

Chapter 3: Basic features of our approach

The task of a syntactic parser of a formal language is clear – to provide all possible syntactic structures for a given input. Formal languages (at least those which are designed for practical applications, not for theoretical purposes) are usually designed in a manner which tries to make this task only as difficult as it is really necessary. Typically there are no ambiguous input symbols, the languages may be parsed by relatively standard means (they are rarely more complex than context-free languages) and also syntax and semantics are usually clearly separated from each other. A good and practically useful formal language should be designed not only with the aim at its suitability for a given purpose, but also with the aim at easy, fast and unambiguous parsing of that language.

Natural languages, on the other hand, have many properties, which make their parsing very complicated. The input often contains word forms, which are morphologically ambiguous not only in gender, number or case, but also with respect to the part of speech information (cf. in Czech *stát* [to stand/the state], *ženou* [woman(instrumental case)/(they) chase] or *tři* [three/rub(imperative)].

The syntax and semantics of a natural language accompanied by pragmatics and real world knowledge are so closely bound together that in some cases it may mislead even native speakers of a particular natural language. A simple sample sentence supporting this claim was already presented in the first chapter. That sentence may seem to be artificial, so let us give one more example of the complexity of the problem, this time taken from a newspaper:

V novém vládě budou obsazena doposud prázdná křesla.

[Lit.: In new[loc.sg. masc./anim./neut] government[dat./loc.sg.fem.] will be_occupied up_to_now empty chairs.]
(In the new one, the up to now empty chairs will be taken away from the government.)

At first sight this sentence is clearly ill-formed due to the adjective-noun disagreement in gender of the second and third input word. Unfortunately the situation is not so straightforward. A syntactically plausible reading of this sentence is much more difficult to find than in the previous case, but nevertheless it exists. The key to this reading is hidden in the fact that the sentence may contain an ellipsis following the adjective *nový* [new]. The noun *vládě* [government] may then be in the dative instead of the locative case and thus it may have a role of a free modifier of the verb *obsazena* [occupied] instead of being a part of the free modifier expressed by a prepositional nominal group in the locative case (that is the preferred reading from of the original sentence). The free modifier is then expressed by the ellipsis. The acceptability of the syntactically well-formed reading in this case depends on the context and real-world knowledge, but there are several examples of very similar sentences which are definitely acceptable. Let us list one of them:

V novém (divadle) majitelům budou obsazena všechna doposud prázdná křesla.
[Lit.: In new (theatre) to_owners will be_occupied up_to_now empty chairs.]
In the new one, the up to now empty chairs will be taken away from the owners.

Similarly as in this example, if there are any empty chairs belonging to the government somewhere, then the syntactically correct reading may be taken into account. Nevertheless, even in such a case the higher complexity of the syntactically plausible reading will probably play an important role in acceptance or rejection of the sentence. It is highly probable that this sentence will be rejected by a majority of human readers. This example supports our general strategy to concentrate on syntactic parsing and to avoid the involvement of semantics and pragmatics wherever possible. The task of a complete linguistic analysis of natural languages in the sense defined in the first chapter is too complex to be handled in one step. It generally seems to be a good strategy to solve complex problems by dividing them into a series of smaller ones. That is the main reason why we have decided to investigate two main tasks in this dissertation, namely the task of finding one of the possible approaches to the problem of dividing the analysis of a natural language into smaller steps and the task of a thorough investigation of one of those smaller steps, namely the robust syntactic parsing. The problem of semantic or even pragmatic analysis is simply too complex to be handled together with syntactic analysis. The same holds for the incorporation of the real world knowledge into the process of analysis. With the current level of knowledge achieved in the field of knowledge representation we cannot even think about it.

There are, of course, many other differences between formal and natural language parsing, but even those mentioned above provide a ground for asking important questions. We may, for example, ask the question about the strategy of how to cope with the ill-formed input. Should the possibility of the occurrence of a syntactic error in the input text be taken into account? And when: only in case there would be no syntactically correct structure found for a complete input sentence or should the possibility of ill-formedness be taken into account from the very beginning of parsing? The following text does not provide a complete set of answers to these questions, it rather advocates one of the possible approaches to the problem of robust parsing of Czech and discusses its advantages and drawbacks. We also hope that the methods presented here are applicable not only to Czech, but also to other (Slavic) languages with a high degree of word-order freedom.

3.1 An approach to robustness

Another question which to a high degree influences not only the quality of results, but also the complexity of the task itself, is the question of a manner in which the robustness should be incorporated into the parser. It goes without question that the task of robust parsing is very closely related to parsing well-formed input, but the problem is which strategy should be chosen to enrich the classical parsing methods in order to provide acceptable parsing results for ill-formed sentences. This basically means that we have to decide if we are going to build a system composed of two basic parts, both of them separate and more or less independent. The first part would be a standard non-robust parser capable of creating either complete syntactic structures for well-formed (with respect to the grammar of the parser) sentences or a set of (sub)trees representing partial structures of the input sentence. The second part would basically consist in an algorithm attempting to construct syntactic representations of the whole input sentence based on these partial trees.

Another possible approach is to try to use similar methods as in grammar-based (not pattern based) grammar-checking. In this case we will first try to localize and identify a syntactic error (or, more exactly, the syntactic inconsistency) of the input and then to make some "correction" which will allow to create the required syntactic representation of a particular input sentence. In this case the quality of the result to a large extent depends on the accuracy of the error identification, which substantially influences the quality of error-correcting mechanism.

It is quite natural that both approaches have certain advantages and disadvantages and thus every robust parser will probably try to combine to a certain extent both approaches in order to exploit their advantages and suppress their weaker points.

The advantage of the first approach is the separation of a standard parser and its robust component. That allows to use a standard formalism for the description and implementation of the parser and to develop only a certain kind of mechanism for handling partial parses. A similar approach was proposed even for one version of a grammar checker in the early stage of the project LATESLAV [Oliva 93], but it was abandoned later, while the second approach provided a base for a successful implementation of a prototype of a grammar-based grammar checker for Czech (cf. [Holan et al. 97]).

The main disadvantage of the first approach (and also the disadvantage of all other methods based on the evaluation of partial parses) lies in the complexity of the problem of choosing the "correct" partial parses, which may be used for creating a global syntactic structure of a particular sentence. Simple strategies do not seem to be useful and the complexity of more sophisticated methods may exceed the complexity of the original problem. That leads directly to the application of a certain kind of heuristics. The quality of the result is then substantially influenced by the quality of the heuristics used.

The other approach, using a method similar to grammar-based grammar checking, relates the problem of reconstructing the "correct" form of the syntactic representation of a sentence to the problem of localization and identification of syntactic inconsistencies. The difference between a syntactic inconsistency and a syntactic error, as we understand it, is based on the assumption that one particular syntactic error may manifest itself during the parsing process by means of several inconsistencies. A syntactic inconsistency is a particular conflict of two parts of an input sentence, which is encountered during the parsing process. The syntactic inconsistency in fact means that there is no rule in the grammar allowing to join two particular partial structures into a single one. The syntactic error, on the other hand, may be determined only after several syntactic inconsistencies are evaluated. For example, in a Czech sentence *„Mladí chlapec šli domů“* (*Young*[nom. pl] *boys*[nom. sg.] *went*[pl.] *home*) there are two inconsistencies, one is the number disagreement between *mladí* a *chlapec*, the other is the subject-predicate disagreement between *chlapec* and *šli.* Both are caused by the same syntactic error, namely the incorrect form of *chlapec* (instead of *chlapci*).

Without a precise knowledge about the nature and location of syntactic errors it is not possible to build a reliable estimation of a "correct" syntactic tree. This is the main reason why we have decided to use in our system the methods developed in the LATESLAV project (cf. [Kuboň et al. 97]) and to adapt them for the task of robust parsing. As we have already mentioned, one of the main differences between the robust parsing and the grammar checking is the fact that the grammar checker aims at the syntactic recognition of syntactically well-formed sentences, while the robust parser needs to perform a complete syntactic analysis.

3.2 Basic principles of building syntactic trees

In the first chapter we have already defined the dependency trees we are using for the representation of the syntactic structure. The definition describes the data type used for this purpose, but it does not give any information about the manner how the syntactic structure is encoded into the dependency tree. In this paragraph we would like to discuss several basic principles used for coding the syntactic structure of input sentences:

- All input items (word forms, punctuation marks etc.) are represented by separate nodes. This is hardly surprising because, for example, in the Prague Dependency Treebank (a semiautomatically annotated treebank of syntactically tagged Czech sentences [Bémová et al. 97]) the same approach was adopted on the analytical level.
- When assigning the roles of the governing and the dependent nodes to a pair of mutually connected nodes in the syntactic tree we try to follow the linguistic tradition whenever possible, with only a few exceptions. Among these exceptions are those pairs of nodes where the tradition does not provide a clear guideline. This is, for example, the case of coordination inside one clause. If we take a sentence “Petr a Pavel šli do kina.” (Petr and Pavel went to the cinema.), we may use at least six different ways of expressing the coordination in the syntactic tree by means of oblique edges:

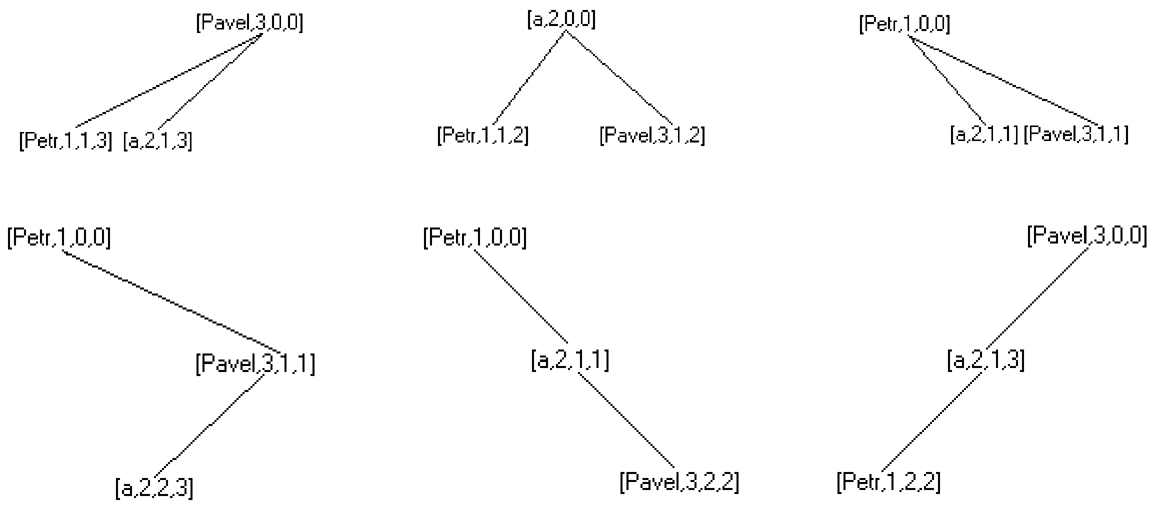

Fig. 3.1. Six variants of expressing coordination by means of oblique edges of the syntactic tree

- In similar cases, when the choice between possible variants is more or less arbitrary, we have decided to use very simple criterion for the decision about which of the nodes should be the governing and which the dependent one. We compare the amount of information stored in individual nodes and assign the role of the governor to the node that contains more morphological or syntactic information. We follow a very simple idea whenever possible – we think that the natural language parsing is a very complex task even if we do not burden it with unnecessary (with a strong stress on this adjective) transport of information through the dependency tree.
  In the example above this criterion clearly rules out the variants with a conjunction *a* [and] as the governing item of at least one pair of nodes (the second, fifth and sixth picture), because the conjunction carries much less information then both nominal nodes. The decision between the remaining three variants is also inspired by our endeavor to make the parsing process as simple as possible. Under normal circumstances it is usually much easier to determine where the coordination ends than where it starts (due to the presence of coordinating conjuction preceding the last member of the coordination), therefore we try to process the coordination from right to left. The last noun and the conjunction are processed first (that rules out the third variant) and the leading noun is attached to the result (the first variant) or vice versa (the fourth variant). The choice between these two variants is more or less arbitrary. We have decided to use the fourth variant of notation throughout this dissertation. According to our opinion this variant is more readable e.g. for coordinated sentences, where the higher depth of the subtree allows to distinguish more easily the two coordinated clauses from each other.
  The general strategy described here applies also to the decision whether a nominal group in prepositional case should depend on a noun or on a preposition and to other similar cases. We prefer the notation with a noun as a governing node and the preposition as the leftmost dependent node of the subtree representing the whole group.
- According to our definition of the dependency tree there are no labels assigned to their edges. This fact is a consequence of our decision to concentrate on creating a structure of the sentence and to leave aside everything that is not substantial. This decision is supported by two facts:
  - The analytical function can be assigned later on the basis of the information about the number of a particular grammar metarule, according to which the edge was created and on the basis of the information contained in the governor or dependent. For example, if the dependent noun is attached to the finite verb by means of a metarule for filling a valency frame slot and if the dependent noun is in the nominative case, then it is highly probable that it is the subject of the clause (there are only very few examples of constructions in Czech violating this rule, e.g. *Jako šéf se Milan změnil.* [As a boss Milan changed.] where both *Milan* and *boss* are in nominative case).
    The connection of analytical functions and individual metarules of the grammar of the system is discussed in more detail in Chapter 9 describing individual metarules.
  - The experiments were made in the past whether it is possible to assign analytical functions reliably if we know the syntactic structure of the sentence [Starý 97]. These experiments used stochastic methods for choosing the proper analytical function and they achieved very good results.

3.3 Reduction of the number of results of syntactic parsing

If we decide to aim at building syntactic trees on the basis of purely syntactic and morphological information only, we also have to make some important decisions before starting to implement the grammar. The omission of semantics affects the number of possible variants of syntactic trees – without semantic clues it is in certain cases impossible to decide to which node of the syntactic tree to attach a dependent subtree. The number of possible readings of the input sentence thus grows. Let us consider the following pair of sentences:

(a) *Banky snižují úroky z ekonomických důvodů.*
 (Banks are lowering interests for economical reasons)

(b) *Banky snižují úroky z krátkodobých půjček.*
(Banks are lowering interests of short-term loans)

When we take into account the meaning of both sentences, we will end up with one syntactic tree representing the structure of each sentence:

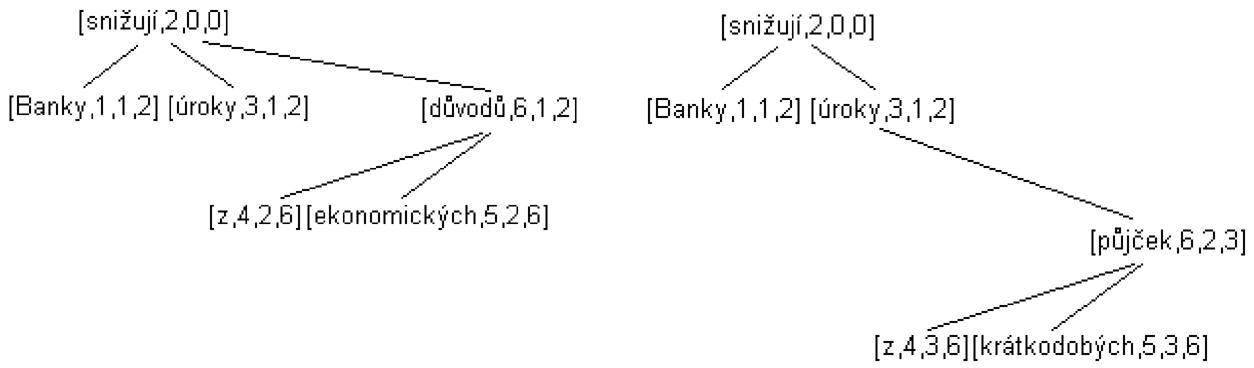

Fig. 3.2. Syntactic trees representing preferred readings of sentences (a) and (b), respectively

If semantics is not taken into account, it is impossible to distinguish between these two sentences and it is necessary to issue for each of the sentences both types of trees.

The previous example demonstrates that the decision to omit semantics and to concentrate on pure syntax has an influence on the number of structures obtained as a result of parsing. Semantics or pragmatics is often very important, for example, in assigning the “correct” free modifiers to verbs or nouns. Without semantic or pragmatic clues it is necessary to consider the combinations of all free modifiers in all possible manners (a nice example of a very complicated structure of free modifiers is the sentence (2) of the testbed presented in Chapter 11). In order to avoid unnecessary duplicity of results we have decided to reduce the number of results generated by our parser by means of the application of general constraints blocking the multiple attachment of a single free modifier. We will discuss these constraints in more detail in Chapter 9 describing the particular metarules of our system (namely in section 9.2 Verbal modifiers, in the description of the metarule(6)). One example of such a constraint is the decision to attach the free modifier (or, more exactly, a nominal group in a prepositional case) to the nearest candidate from the left-hand side (in the sense of the linear order of nodes given by their horizontal index). For the sample sentences (a) and (b) we will get the following D-trees:

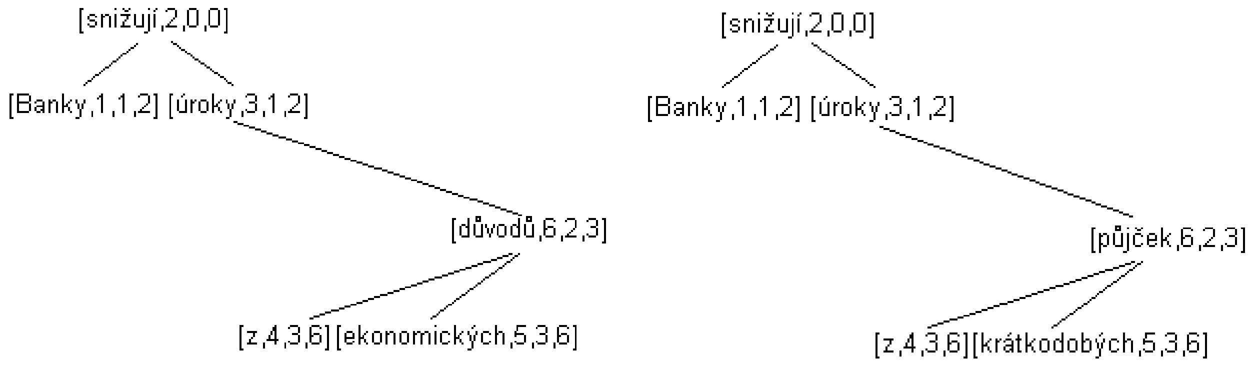

Fig. 3.3. Syntactic trees representing results of syntactic parsing of sentences (a) and (b), respectively, by means of our robust parser

This may seem to be too much simplified, but in fact, no information is lost. If we take into account that the general constraint forbidding the attachment to any other than the nearest preceding candidate was applied, we may easily reconstruct all remaining results. The structures obtained give us a very good clue which nodes belong among the candidates, so we may generate a full set of results. The fact that the free modifier may be attached also to the main verb of the sentence leads to the following pair of results for our sample sentences:

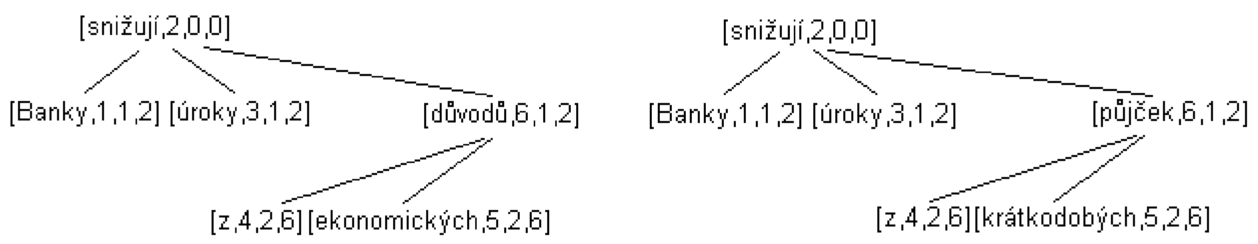

Fig. 3.4. Syntactic trees representing those structures for sentences (a) and (b), respectively, which can be generated from structures in Fig. 3.2

The application of general constraints in certain metarules thus allows us to create a subset of results of a syntactic parser, which represents the whole set in the sense that, if required, the full set of results may be obtained in a simple and regular way out of the subset. In this sense we speak about a *skeleton* of the set of possible syntactic trees. A thorough discussion of the problem of generating additional syntactically plausible structures to a single structure representing the preferred reading of a sentence may be found in [Panevová, Straňáková 99], therefore we are not going to cover the problem here. The authors take as a basic structure the trees from the Prague Dependency Treebank (there is only one tree for each sentence in the PDT representing the most likely reading from the point of view of a human annotator) and formulate a set of rules allowing to generate all syntactically plausible variants of dependency trees for each sentence.

The ambiguity of attachment of prepositional groups is very common in Czech. In most cases these groups play a role of free modifiers, but quite frequently they may even belong to a set of inner participants of the governing word. One of the major problems of syntactic parser is its inability to distinguish these two cases from each other. The endeavor to develop sophisticated rules solving this problem is described for example in [Straňáková 99] or [Straňáková 01]. Even more problematic is the robust parsing of ill-formed sentences. The task of the proper free modifier attachment is complex enough even in case it is possible to rely on the fact that the input sentence is well-formed. The fact that we should expect ill-formed input sentences means that there is even less information, which may be used for resolving the ambiguity of the free modifier attachment.

These reasons led us to a substantial simplification of valency frames (compared to the theoretical version described for example in [Panevová 80]. Due to our inability of distinguishing for prepositional groups whether a particular group is a free modifier or an inner participant it is useful to handle these two types of valency frame members in a uniform way. As our simplified valency frames do not contain any slots for nominal groups in prepositional cases, we always attach nominal groups in prepositional cases as free modifiers. The structure of valency frames used in our robust parser is described in a more detail in Chapter 8.

Chapter 4: Basic components of the system

Every project of natural language parsing may be divided into a number of smaller specialized subsystems. Among the most important ones there are, for example, a morphological analysis and disambiguation of the input (very important for languages with a rich inflection), a parsing formalism, a grammar, a dictionary capable to provide information required for parsing. Equally important are the tasks that do not constitute a proper part of the system, but provide a valuable support in the phase of system implementation. The task of the development of a method for effective testing and debugging the grammar including the development of a representative test suite belongs to this category. Let us now introduce these units and tasks and divide them into two categories – already existing resources, which were reused for our purposes, and those which had to be created anew.

4.1 Available resources

It is always a great advantage if a new project can reuse the already existing resources. Also for our implementation of the robust parser for Czech it was not necessary to build all of its components from scratch. Some of the important resources were already available and were reused. Let us briefly mention these resources:

4.1.1 Morphology

The morphological analysis of Czech had already been created in the past [Hajič 94], originally for a commercial spelling checker of Czech. The morphological dictionary now covers almost the entire Czech vocabulary. There is nothing that could be added in the frame of this project, so the problem of morphological analysis of Czech was not addressed here at all. We can skip the phase of morphological analysis altogether supposing that every input sentence has already been processed by morphological analysis and that it already carries all relevant morpho-lexical information as part of the input.

4.1.2 Software environment

The choice of software environment for the development of the parser is a decision that to a great extent influences the general behavior of the system. There is usually a trade-off between the speed and efficiency and the use of a high-level linguistic formalism. Especially unification-based formalisms, probably the most widely used tool in current computational linguistics, are infamous for a great inefficiency of their implementations. Fortunately, the RFODG (Robust Free Order Dependency Grammar) developed in the LATESLAV project [Kuboň et al. 97] together with FODG (Free Order Dependency Grammar) [Holan et al. 00] proved to be useful even in the project of a robust parser for Czech. The main advantage of the RFODG is its ability to handle syntactically correct and ill-formed sentences in a uniform way, while FODG already proved its suitability for the description of nonprojective constructions and different kinds of measures allowing to formulate general constraints applicable in robust parsing.

4.2 New components

4.2.1 Grammar

There are at least two reasons why it is not possible to use a context-free grammar directly for natural language parsing. The first reason is that context-free grammar is too weak for an adequate description of Czech (cf. the discussion of parsing complexity of nonprojective constructions in Chapter 6), the second reason is that it is also impractical – it would simply require too many rules. It is therefore quite natural to use a metagrammar instead of a context free grammar. Such a metagrammar is in fact a two-level grammar – each metarule of the metagrammar represents a finite set of grammar rules. The metarule also allows us to use a (finite) set of attributes and it should also provide means for formulating constraints. There might generally be two kinds of constraints – the first kind are the constraints imposed on values or types of individual attributes, the second kind are general constraints controlling the application of individual metarules. The metagrammar also allows pruning spurious or inadequate structures. Chapter 9 contains a detailed description of our metagrammar and its metarules.

The metagrammar is, of course, far from complete in the sense that it does not cover all subtle syntactic phenomena of Czech, but it covers the most frequent ones, which allow us to represent the syntactic structure of simple clauses and also the structure of certain types of complex sentences. It also contains a few examples of how marginal phenomena should be handled in the future. The grammar is important from at least one more point of view – it provides a basis for the discussion of the influence of certain linguistic phenomena on parsing complexity and efficiency. It also serves as a background for the discussion of possible improvements of the methodology and technology of the robust parser leading to a more adequate and efficient implementation of the robust parser in the future.

4.2.2 Dictionary

In order to syntactically parse an input sentence it is, of course, necessary to combine the morpho-lexical information acquired in the phase of morphological analysis with the lexico-syntactic information contained in the syntactic dictionary of the system. The task of building such a syntactic dictionary is very costly and time–consuming, therefore we use in this project only a sample of such a dictionary. The dictionary structure used in the LATESLAV project [Skoumalová 94] was modified and a sample dictionary containing a representative selection of lexical items was developed for the purpose of the robust parser.

4.2.3 Testing and debugging

The last but not least important task addressed also in this project was the development of a method allowing for the effective testing and debugging of the grammar during the process of its development. One of the most substantial problems of nondeterministic approach to parsing is the problem of preserving the consistency of new and modified rules of the formal grammar with the already existing rules. It is very often the case that the newly added rule to a certain extent overlaps with one or more already existing rules and it is necessary to specify a set of constraints adjusting the relationship between these rules. It is one of the crucial problems of every project aiming at a development of a large-scale formal grammar of a natural language. The problems of testing and debugging are discussed in Chapter 10.

Chapter 5: Basic notions II

One of the most useful results of the LATESLAV project was the introduction of a class of formal grammars called RFODG (Robust Free-Order Dependency Grammars). This class of formal grammars is based on the earlier definition of dependency grammars related to formal languages, which can be found for example in the textbook [Gladkij 73]. Our definition was developed as a tool for the description of syntactically ill-formed sentences of a language with a high degree of word-order freedom and for the differentiation of syntactically well-formed and ill-formed sentences. It also provides means for localization of syntactic inconsistencies. The exact definition of RFODG can be found in [Kuboň et al. 97]. The following paragraphs contain only that part of the description of RFODG which is necessary for the sake of understanding the remaining chapters.

5.1 Robust Free Order Dependency Grammar

RFODG serves primarily for the description of surface syntax. It provides the base of the parsing with subsequent localization and evaluation of syntactic inconsistencies and errors (as already mentioned, it is useful to distinguish between the notion of syntactic inconsistency as an instance of a violation of a syntactic rule and the notion of syntactic error as a result of evaluation of one or more syntactic inconsistencies). It is assumed that RFODG is applied when the lexical and morphological analysis is already completed, it was not designed as a formalism for error detection on lexical level. In the sequel the RFODG's are analytic (recognition) grammars.

It is also assumed that the result of lexical and morphological analysis for each word form of the input is a finite set of symbols representing lexical and morphological properties of the word form. It is further assumed that formal syntax is connected with lexical analysis in that the terminal symbols of the formal grammar accounting for syntax are the symbols representing lexical and morphological properties of the word forms.

The RFODG uses the following types of classification of the set of symbols:

a) terminals and other symbols (nonterminals)

b) positive and negative symbols.

The sets under a) have the usual meaning. The terminals of RFODG are the lexical categories of the morpho-lexical analysis combined with the lexico-syntactic information (valency frames etc.).

The sets under b) serve for the localization and rough classification of syntactic inconsistencies.

Each pair of sets of symbols under a) and b) constitutes the set of all symbols used by RFODG. That means that each symbol of the set of symbols V (cf. Definition 5.1) is at the same time a member of exactly two subsets, one from each pair.

Definition 5.1. (of the RFODG)

Robust Free-Order Dependency Grammar (RFODG) is a 4-tuple $(\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{St}, \boldsymbol{P})$, where $\boldsymbol{N}$ is the set of nonterminals, $\boldsymbol{T}$ is the set of terminals and the union of $\boldsymbol{N}$ and $\boldsymbol{T}$ is denoted as $\boldsymbol{V}$, $\boldsymbol{St} \subset \boldsymbol{N}$ is the set of root symbols (starting symbols), and $\boldsymbol{P}$ is the set of rewriting rules of two types of the form:

a) $\boldsymbol{A} \rightarrow_{\mathrm{X}} \boldsymbol{BC}$, where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \boldsymbol{V}$, $\boldsymbol{X}$ is denoted as the subscript of the rule, $\boldsymbol{X} \in \{\boldsymbol{L}, \boldsymbol{R}, \boldsymbol{LP}, \boldsymbol{RP}\}$,

b) $\boldsymbol{A} \rightarrow \boldsymbol{B}$, where $\boldsymbol{A}, \boldsymbol{B} \in \boldsymbol{V}$.

We suppose that $\boldsymbol{V} = \boldsymbol{V_p} \cup \boldsymbol{V_n}$, where $\boldsymbol{V_p}$ is the set of positive (correct) symbols, and $\boldsymbol{V_n}$ is the set of negative symbols (negative symbols mark syntactic inconsistencies contained in the tree representing the syntactic structure of a sentence).

The occurrence of the letter ***L*** in the subscripts of the rules means that the first symbol on the right-hand side of the rule is considered *dominant*, and the other *dependent*.

The occurrence of the letter ***R*** in the subscripts means that the second symbol on the right-hand side of the rule is considered *dominant*, and the first one *dependent*.

If a rule has only one symbol on its right-hand side, the symbol is considered as *dominant*.

Applying such a rule means to rewrite an occurrence of the right-hand side symbol by the left-hand side symbol.

A rule whose right-hand side contains two symbols is applied (for a reduction) in the following way:

The dependent symbol is deleted, and the dominant one is rewritten (replaced) by the symbol standing on the left-hand side of the rule.

The rules $\boldsymbol{A} \rightarrow_{\mathrm{L}} \boldsymbol{BC}$, $\boldsymbol{A} \rightarrow_{\mathrm{R}} \boldsymbol{BC}$ can be applied for a reduction of a string ***z*** to any of the occurrences of symbols ***B,C*** in ***z***, where ***B*** precedes ***C*** in ***z***.

This in fact means that in RFODG it is possible to apply this type of rules on discontinuous pairs of symbols from ***z*** (*immediate* precedence of ***B*** is not required)**.** This property of RFODG is used for capturing nonprojective constructions.

The rules $\boldsymbol{A} \rightarrow_{\mathrm{LP}} \boldsymbol{BC}$, $\boldsymbol{A} \rightarrow_{\mathrm{RP}} \boldsymbol{BC}$ can be applied for a reduction of a string ***z*** to any *neighboring* occurrences of symbols ***B,C*** in ***z***, where ***B*** precedes ***C*** in ***z***.

The definition allows the terminals to appear on the left-hand side of rules. This is due to the fact that we consider this grammar to be an analytical grammar. The analysis is done by reduction. If, for example, a sequence of reductions analyzing the subordinate clause is completed, it is necessary to represent the subordinate clause by an appropriate terminal symbol on the level of the governing clause in order to maintain the well-formedness (or ill-formedness) of a given sentence for the purpose of further analysis.

For the sake of the following explanations it is necessary to introduce a notion of a *DR-tree* (delete-rewrite-tree) according to *G*. A *DR-tree* maps the essential part of history of deleting dependent symbols and rewriting dominant symbols performed by the rules applied. The definition of a *DR-tree* was for the first time introduced in [Holan et al, 1998].

Put informally, a *DR-tree* (created by a RFODG *G*) is a finite tree with a root and with the following two types of edges:

vertical (V-edges): these edges correspond to the rewriting of the dominant symbol by the symbol which is on the left-hand side of the rule (of *G*) used. The vertical edge leads (is oriented) from the node containing the original dominant symbol to the node containing the symbol from the left-hand side of the rule used.

oblique: these edges correspond to the deletion of a dependent symbol. Any such edge is oriented from the node with the dependent deleted symbol to the node containing the symbol from the left-hand side of the rule used.

Let us now provide a more formal definition of DR-trees. In the sequel the symbol *Nat* denotes the set of natural numbers (without zero):

Definition 5.2.(DR-tree)

A triple *Tr=(Nod,Ed,Rt)* is called *DR-tree* created by a RFODG *G* (where *Nod* means the set of nodes, *Ed* the set of edges, and *Rt* means the root node), if the following points hold for every $U \in$ *Nod*:

a) *U* is a 4-tuple of the form [A,i,j,e], where $A \in V$ (terminal or nonterminal of *G*), $i,j \in Nat$, *e* is either equal to *0* or it has the shape k_p, where $k,p \in Nat$. The *A* is called *symbol of U*, the number *i* is called *horizontal* index of *U*, *j* is called *vertical index*, *e* is called *domination index*. The horizontal index expresses the correspondence of *U* with the i-th input symbol. The vertical index corresponds to the length increased by 1 of the maximal path leading bottom-up to *U*. The domination index either represents the fact that no edge starts in *U* (*e=0*) or it represents the final node of the edge starting in *U* (*e=k_p*, cf. also the point e) below).

b) Let *U= [A,i,j,e]* and *j>1*. Then there is exactly one node U_1 of the form *[B,i,k,i_j]* in *Tr*, such that *1 <= k < j* and the pair *(U_1, U)* creates a vertical edge (V-edge) of *Tr*, and there is a rule in *G* with *A* on its left-hand side, and with *B* in the role of the dominant symbol of its right-hand side.

c) Let *U= [A,i,j,e]*. Then *U* is a leaf if and only if $A \in T$ (terminal symbol of *G*), and *j=1*.

d) Let *U= [A,i,j,e]*. *U=Rt* iff it is the single node with the domination index (*e*) equal to 0.

e) Let *U= [A,i,j,e]*. If $e=k_p$ and $k < i$ (resp. $k > i$), then an oblique edge leads from *U* (dependent node) to its mother node U_m with the horizontal index *k*, and vertical index *p*. Further a vertical edge leads from some node U_s to U_m. Let *C* be the symbol from U_m, *B* from U_s, then there exists a rule in *G* of the shape $\boldsymbol{C} \rightarrow_L \boldsymbol{BA}$ (resp. $\boldsymbol{C} \rightarrow_R \boldsymbol{AB}$).

f) Let *U= [A,i,j,e]*. If $e=k_p$, and *k=i* , then a V-edge leads (bottom up) from *U* to its mother node $U_m = [B,i,p,e_m]$ (for some e_m). If there is not such an oblique edge, for which U_m is its dominating node, then there exists a rule in *G* of the shape $\boldsymbol{B} \rightarrow \boldsymbol{A}$. In the other case, see the point e).

We say that a *DR-tree Tr* is *complete* if for any of its leaves *U= [A,i,1,e]*, where *i>1*, it holds that there is exactly one leaf with the horizontal index *i-1* in *Tr*.

Example 5.1.

This example illustrates the notion of *DR-tree*. Let us take the sentence from the example 1.2:

Zadaný soubor se nepodařilo otevřít.

[Specified file Refl. it_was_not_possible to_open.]

(The specified file failed to open.)

The following grammar G_1 is a RFODG. $G_1 = (N_1,T_1,\{S\},P_1)$, T_1 = {Zadaný, soubor, se, nepodařilo, otevřít, . }, N_1 = {N,V,I,S}, P_1 = {N->$_R$ Zadaný soubor, V->$_R$ se nepodařilo, V->$_L$ V I, I->$_R$ N otevřít, S->$_L$ V .}. Fig. 5.1 displays a DR-tree Tr_1 derived by G_1 for the sentence. The nodes of Tr_1 are:

L_1=[Zadaný,1,1,2_2], L_2=[soubor,2,1,2_2], L_3=[se,3,1,4_2], L_4=[nepodařilo,4,1,4_2], L_5=[otevřít,5,1,5_3], L_6=[.,6,1,4_5] (leafs) and N_1=[N,2,2,5_3], N_2=[V,4,2,4_4], N_3=[I,5,3,4_4], N_4=[V,4,4,4_5] and N_5=[S,4,5,0].

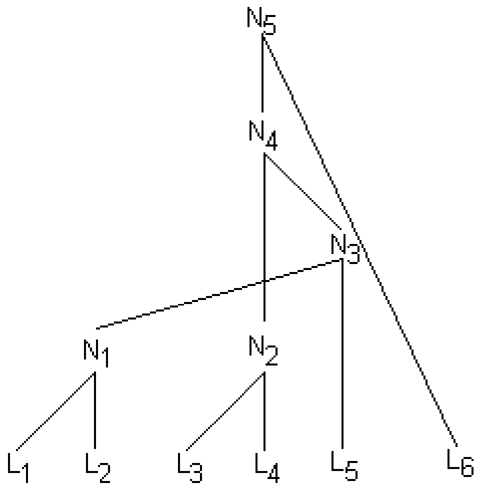

Fig. 5.1. A DR-tree Tr_1 derived by G_1 for the sentence *Zadaný soubor se nepodařilo otevřít.*

Definitions 5.3.

TN(G) denotes the set of complete DR-trees rooted in a symbol from S_t, created by G. If Tr $\in$ TN(G), we say that Tr is parsed by G.

Let $w=a_1a_2 ... a_n$, $w \in T^*$, $Tr \in TN(G)$, and let [a_i,i,1,e(i)] denote the i-th leaf of *Tr* for *i=1, ... ,n*. In such a case we say that the string *w is parsed into Tr* by *G*.

The symbol *L(G)* represents the set of strings (sentences) parsed into some *DR-tree* from *TN(G)*.

We will also write *TN(w,G)={Tr; w is parsed into Tr by G}*.

Remark

At this point it is necessary to stress that the RFODG is not implemented directly – the grammar composed of rules of the type defined above would be too large. In order to avoid this obstacle we have decided to develop a special software environment which allows describing the grammar by means of metarules representing sets of related rules of the RFODG. This "metalanguage" is described in a more detail in Chapter 7.

5.2 A relationship between DR-trees and D-trees

There is a very straightforward correspondence between DR-trees and D-trees (as they were defined in the first chapter). Informally, a D-tree is obtained by the contraction of vertical paths of a DR-tree and by the addition of information about the distance of a particular node from the root (vertical index).

Definition 5.4. (A tree contracted from a DR-tree)

Let *Tr* ∈ *TN(w,G)* (w is parsed into *Tr* by G), where w=a_1a_2 ... a_n. The *tree dT(Tr) contracted from the DR-tree Tr* is defined as follows: The set of nodes of *dT(Tr)* is the set of 4-tuples [a_i*,i,j(i),k(i)*] (a_i is the i-th symbol of *w*). For each symbol a_i there is exactly one node in *dT(Tr)*.

k(i)=0 if and only if the root of *Tr* has the horizontal index *i* (then the [a_i*,i,j(i),k(i)*] is also the root of dT(Tr)).

k(i) ∈ *Nat* if and only if in *Tr* an oblique edge leads from some node with the horizontal index *i* to some node with the horizontal index *k(i)*.

The index *j(i)* is constructed in several steps, starting with the node representing the root of *dT(Tr)*:

1. *j(i)* is assigned the value *0* if and only if the root of *Tr* has the horizontal index *i* (then the [a_i*,i,j(i),k(i)*] is also the root of dT(Tr));
2. *j(i)* is assigned the value *1* if and only if there is an oblique edge in *Tr* leading from a node with the horizontal index i to the node with the same horizontal index as the root.
3. *j(i)* is assigned the value *n* if and only if there is an oblique edge in *Tr* leading from a node with the horizontal index i to the node with the same horizontal index as a node [a_m*,m,n-1,k(m)*] ∈ *dT(Tr)*.

The third slot of nodes of *dT(Tr)* is called the *vertical index.*

We can see that the edges of *dT(Tr)* correspond (one to one) to the oblique edges of *Tr*, and that they are fully represented by the second and the fourth slot of nodes of *dT(Tr)*. The second slot is called the *horizontal index* of the node. The fourth slot is called the *dominance index*.

Example 5.2.

Let us take the sentence from the previous example and let us contract the DR-tree from Fig.5.2.

Zadaný soubor se nepodařilo otevřít.

[Specified file Refl. it_was_not_possible to_open.]

(The specified file was the one it was not possible to open.)

There are the following nodes in the contracted tree dT(Tr) of this sentence:

u_1=[Zadaný,1,3,2], u_2=[soubor,2,2,5], u_3=[se,3,1,4], u_4=[nepodařilo, 4,0,0], u_5=[otevřít,5,1,4], u_6=[.,6,1,4]

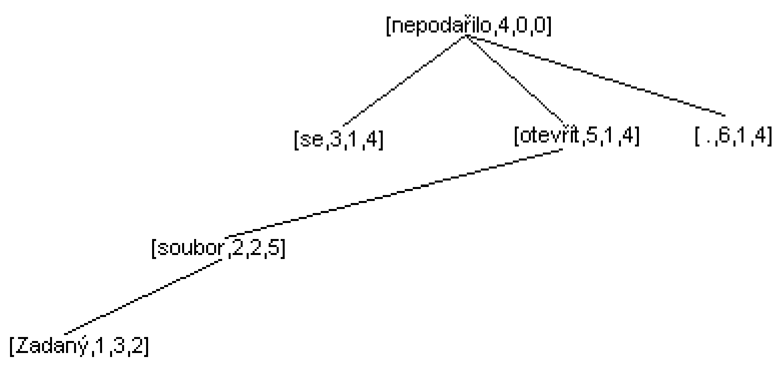

Fig. 5.2. A contracted tree *dT(Tr)* for the sentence from example 5.2

Lemma 5.1.

Let $Tr \in TN(w,G)$ (w is parsed into *Tr* by G), where $w=a_1a_2 \ldots a_n$. The tree *dT(Tr)* contracted from the DR-tree *Tr* is a D-tree.

Proof: *dT(Tr)* contains exactly n nodes, one for each symbol a_i, with the horizontal index $i \in \{1..n\}$. One of these nodes (the contracted root of *Tr),* the node $[a_i,i,0,0]$, is a root of *dT(Tr).* The construction of the *vertical index* of *dT(Tr)* guarantees that its value represents the distance of the node from the root . The dominance index of a node from *dT(Tr)* represents the horizontal index of its governor, as it is described in the definition of dependency trees from chapter 1. If $u=[a_i,i,v_i,d_i]$ and $d_i \neq 0$, then there is exactly one node $v=[a_{d_i},d_i,v_i\text{-}1,d_{d_i}]$. It is the node representing the node from *Tr* to which leads an oblique edge from a node of *Tr* with the horizontal index *i*. For every horizontal index *i* of a node of *Tr* (except the horizontal index of the root) there is only one oblique edge leading from a node with this index to a node with a different horizontal index.

Definition 5.5

If a DR-tree $Tr \in TN(w,G)$ (w is parsed into *Tr* by G) contains only positive symbols (from V_p), we say that *Tr* is *positively parsed* by *G*. If *Tr* contains also negative symbols (from V_n), we say that *Tr* is *robustly* parsed by *G.*

We say that a sentence *w* is positively parsed, if there is a positively parsed DR-tree among the DR-trees parsed from *w*. A sentence *w* is *robustly parsed*, if it is parsed, but not positively parsed.

5.3 Syntactic analysis and a nonprojectivity measure of a language

In this section we would like, first, to generalize the definition of magnitude of nonprojectivity of the pair [sentence; (one of) its structures] to the magnitude of nonprojectivity of a sentence (i.e. without respect to a particular structure of the sentence). Based on this, we define the measure of nonprojectivity for a set of sentences, that is, for a language.

Let us first define the notion of a *D- analysis*, which allows for making statements about a sentence jointly with all its structures (that is, basically, irrespective of the set of its structures).

Definition 5.6 (D-analysis)

Let *A* be a (finite) alphabet and let *T* be a set of dependency structures (trees) over sentences created over *A*. A function *D: A* -> P(T)* is called[1] a *D-analysis* over the alphabet *A*.

That is, a D-analysis is a function[2] which assigns each sentence *w* a set of dependency trees *D(w)*.

[1] P(T) denotes the set of subsets of T

[2] At this moment we do not intend to study the way this function should be defined. We suppose that this function is given by the linguistic knowledge.

In cases where it is clear what the alphabet is, we abbreviate the wording of the term accordingly, and instead of a D-analysis over the alphabet *A,* we speak about D-analysis only.

Let us now introduce the terms *language* and *D-analysis of a language*.

Definition 5.7 (Language)
A *language* over an alphabet *A* is a set $L \subset A^*$. If it is clear what the alphabet is, we abbreviate the term to *language* only.

Definition 5.8 (D-analysis of a language)
Let *L* be a language over the alphabet *A* and let *D* be a (fixed) D-analysis over *A*. We say that *D* is a *D-analysis of the language L*, if the following holds:

$L = \{w \in A^*;\ D(w) \neq \emptyset\}$

that is, the language *L* contains sentences for which the analysis gives at least one tree as a result, and only these sentences.

Having defined all necessary notions, we can finally define the magnitude of non-projectivity of a sentence and of a language.

Definition 5.9 (Magnitude of non-projectivity of a sentence, magnitude of non-projectivity of a language)
Let *L* be a language, $w \in L$ be a sentence and let *D* be a (fixed) D-analysis of *L*. The measure $dNh(w,D) = max_{T \in D(w)}\, dNh(T)$ is called the *magnitude of non-projectivity* of the sentence *w* according to D-analysis *D*.

Let *L* be a language and let *D* be a (fixed) dependency analysis of the language *L*. Let us define $dNh(L,D) = \sup_{w \in L} dNh(w,D)$. The measure *dNh(L,D)* is called the *magnitude of non-projectivity* of the language *L* according to D-analysis *D*.

There is one more measure, which is important for the description of structures obtained as a result of robust parsing and for the formulation of global constraints on parsing. This measure describes the degree of "ill-formedness" of a particular sentence. For this purpose we will need slightly modified D-trees. The original definition of D-trees does not allow keeping track of syntactic inconsistencies encountered in the parsing process. There is a very natural way of adding this information to standard D-trees, namely adding information about the location of syntactic inconsistencies to D-trees.

Definition 5.10 (Robust D-trees)
Let $Tr \in TN(w,G)$ (w is parsed into *Tr* by G), where $w = a_1 a_2 \ldots a_n$. Let *dT(Tr)* be a D-tree contracted from the DR-tree *Tr.* We say that the D-tree *dTr(Tr)* is a *robust D-tree* if it has the following properties:

a) The set of nodes of *dT(Tr)* is the set of 5-tuples $[a_i,i,b_i,j(i),k(i)]$ (a_i is the i-th symbol of *w*). For each symbol a_i there is exactly one node in *dT(Tr).*
b) $k(i)=0$ if and only if the root of *Tr* has the horizontal index *i* (then the $[a_i,i,b_i,j(i),k(i)]$ is also the root of dT(Tr)). $k(i) \in Nat$ if and only if in *Tr* an oblique edge leads from some node with the horizontal index *i* to some node with the horizontal index *k(i)*. The index *j(i)* is constructed in the same manner as in the definition 5.4.
c) The symbol $b_i \in \{0,1\}$ has the value 1 iff in the DR-tree *Tr* the node with the horizontal index *k(i)* towards which the oblique edge from the node with the horizontal index *i* leads contains a negative symbol (nonterminal). It equals 0 in the opposite case or in case that $k(i)=0$. The symbol b_i is called the *index of robustness.*

We can see that the edges of *dT(Tr)* correspond (one to one) to the oblique edges of *Tr*, and that they are fully represented by the second and the fifth slots of nodes of *dT(Tr)*. The second slot is called the *horizontal index* of the node. The fourth slot of nodes of *dT(Tr)* is called the *vertical index.* The fifth slot is called the *dominance index*.

Remark

The definition shows that the robust D-trees are in fact D-trees with the added information about the existence of syntactic inconsistencies. In the trees drawn by the interpreter of our metagrammar as a result of parsing (cf. Chapter 11) are the third slots of all nodes (the index of robustness) represented by edges leading up from a particular node. If the index of robustness of that node equals 1, the path leading up to the dominating node is drawn as a dashed line.

Definition 5.11 (Degree of robustness of DR-trees and D-trees)

Let $Tr \in TN(w,G)$ and T is a *robust D-tree* over a sentence w. We say that a node of Tr is negative, if it contains a negative symbol (nonterminal). We denote as *Rob(Tr)* the number of negative nodes in Tr and *Rob(T)* the number of nodes in T in the form $[a_i,i,1,j(i),k(i)]$ (the nodes with the index of robustness equal to 1). We say that *Rob(Tr)* and *Rob(T)* is the *degree of robustness* of Tr and T, respectively.

5.4 The mutual relationship of nonprojectivity measures and the degree of robustness

As we have already mentioned in previous chapters, we understand the robust parser as a tool for assigning (a) syntactic structure(s) to ill-formed sentences which do not violate certain global constraints and also as a tool for filtering out the sentences which violate them. In this way we would like to simulate the fact that certain ill-formed sentences may be parsed (and understood) by a human, while others are so corrupted that even a human is not able to parse them. This task is, of course, very difficult, because the human has at his or her disposal not only syntax, but also semantics, pragmatics and real world knowledge. That in fact means that we are looking for as natural constraints as possible in order to achieve a goal which would be at least close to the results achieved by humans.

The measures we have defined in this chapter represent one of the possible global constraints which may be used for a robust parser. The degree of robustness and the nonprojectivity measures capture the complexity of sentences relevant from the point of view of a robust parsing of languages with a high degree of word-order freedom. They provide a theoretical base for the division of a parsing process into phases, whose aim is to analyze input sentences step by step, with more relaxed global constraints for each subsequent phase. In our implementation of the robust parser for Czech we have decided to use the following phases:

The first phase, called a *positive projective* phase, tests whether the set of projective positively parsed trees is empty. If not, the sentence is syntactically correct and the parser issues all trees representing the syntactic structures of all syntactically acceptable readings of the sentence.

If the sentence is not projective or is syntactically ill-formed, the second phase, called *negative projective or positive nonprojective,* starts. It tries to find either a positively parsed nonprojective tree or a projective tree with the degree of robustness equal to or greater than one, i.e. a projective tree representing an ill-formed sentence. If the parser succeeds, it stops and issues all relevant trees as the result of parsing.

If the second phase fails, the partial results are handed over to the third phase, which is called *negative nonprojective.* It tries to find a nonprojective tree with the degree of robustness equal or greater than zero. This phase is the last one and if it fails, the whole parsing fails and issues a relevant error message. The practical experiments have shown that even though Czech is a language with an unlimited magnitude of nonprojectivity, cf. [Holan et al 00], it would be reasonable to restrict the magnitude of nonprojectivity to the lowest possible threshold, i.e. one gap. The chapter dealing with actual results of robust parsing of sentences from our testbed clearly shows that the increase of parsing time in each phase is really substantial. We would like to stress that the main task of our work is to develop a method capable of handling the problem of robust parsing, not to concentrate on detailed practical solutions. At this point it is also necessary to point out that a much subtle scale of phases is currently being developed by the author of the software environment used here, Tomáš Holan cf. [Holan 01]. We think that from this point of view our restriction is reasonable.

A combination of negative projective and positive nonprojective parsing in the second phase is quite natural. In some cases it is very difficult or even impossible to find out if the sentence is correct and nonprojective or ill-formed and projective.

We can illustrate this fact by the following sentence:

"Které děvčata chtěla koupit ovoce?"

[Lit.: Which girls wanted [to] buy fruit?]

Some of the native speakers consider this sentence to be correct, some say that it is incorrect and most of them are uncertain, but they usually say that this sentence is simply weird. The problem is that the sentence can be assigned at least three possible trees. Two of them do not contain any syntactic inconsistency (but are nonprojective), with the pronoun *Které* [Which] depending on the noun *ovoce* [fruit] (cf. Fig. 5.2) or on the infinite verb *koupit* [to buy] (cf. Fig. 5.3). The structure from Fig. 5.2. in fact represents two different readings, with the pronoun *Které* either in genitive or in dative case.

At this point we would like to make an important comment concerning the dependency trees in the figures. Their nodes are different from the nodes of dependency trees drawn according to our definition from Chapter 1. The reason is that the figures describing the results of robust parsing are in fact screenshots of actual results obtained. These trees contain complete information necessary for drawing real dependency trees. The symbol a_i represents the whole node, the horizontal index is represented by means of preserving the surface word order from the left to the right, the vertical index is preserved according to the definition as a number of edges leading from the root to a particular node and the dominance index is represented by a particular edge. The presence of a syntactic inconsistency in the tree is represented as a dashed line leading from a governing node to the node with the index of robustness equal to 1. In other words, the dashed line represents an occurrence of a syntactic inconsistency between the nodes belonging to a dashed edge. In some cases the system also draws numbers of metarules, which were used for the creation of a particular edge. The use of simplified nodes was motivated by practical reasons – if a sentence contains more than 20 input words, it is very difficult to use full size nodes and to preserve readability of the structure at the same time. The trees from figures 5.3–5.5 are actual results of our robust parser. The results of our robust parser presented in Chapter 11 are displayed in the same manner, as screenshots of actual results (with simplified nodes).

One more explanation is necessary – all screenshots contain a special node for LEFT_SENTINEL as the governing node of the whole structure. This node is used throughout the input data as a marker showing where the sentence starts.

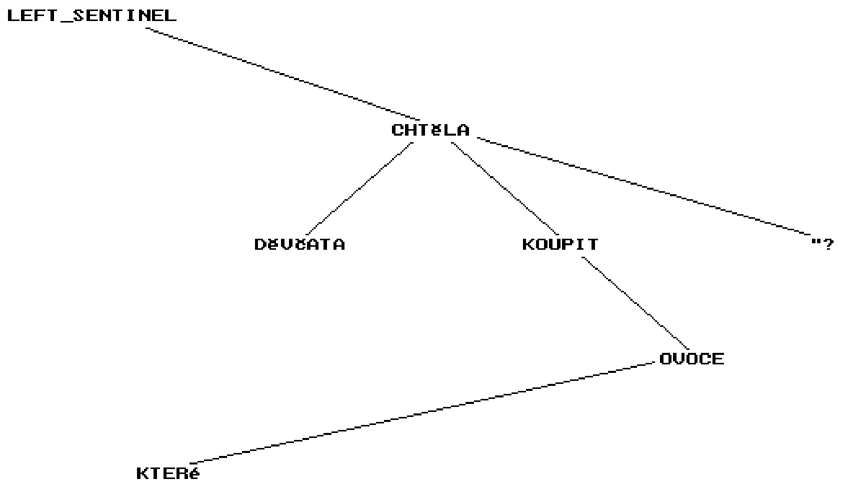

Fig. 5.3. A nonprojective tree representing the sentence “Which fruits did the girls wanted to buy?“

The other reading contains a syntactic inconsistency between the pronoun *Které* [Which] and the noun *děvčata* [girls] – these two words disagree in gender. This reading may be represented by the projective tree from the Fig.5.5.

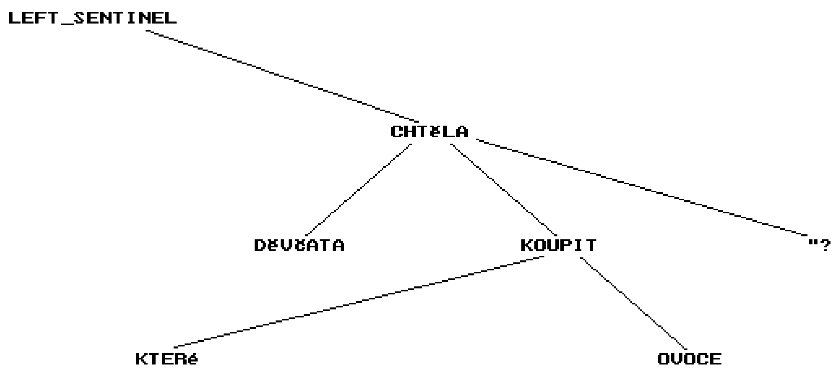

Fig.5.4. The second nonprojective parse representing the sentence “To whom did the girls want to buy fruits?”

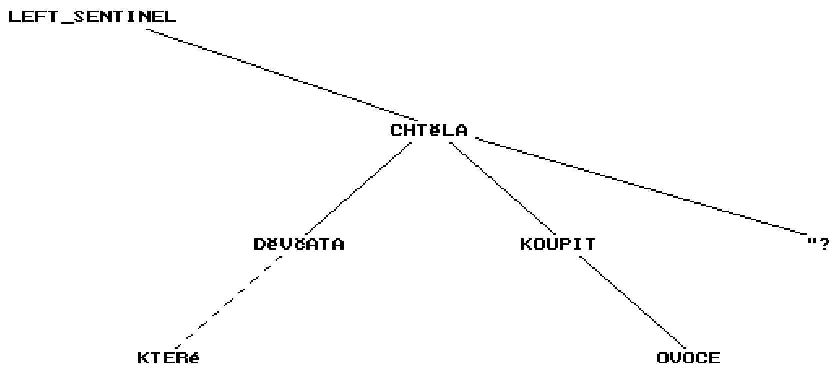

Fig. 5.5. A projective parse with one syntactic inconsistency

It is possible to argue that a nonprojective parse should be preferred, but unfortunately there is not enough of quantitative data to support this claim. Anyway, if such a need arises, there is no problem in dividing the parser into four basic parts, *positive nonprojective* being the second and *negative projective* being the third. We have to be very careful in making this kind of preference, because it has to be taken into account that sometimes a sentence which is considered to be syntactically incorrect by a majority of native speakers may have a more or less obscure parse which is correct from the point of view of formal surface syntax. As we have already stressed, the human speakers usually take into account not only the formal syntax of the sentence when they decide about the grammatical correctness or incorrectness.

Chapter 6: Complexity estimations

In the previous chapter we have mentioned the necessity to limit the degree of nonprojectivity if we want to reduce the complexity of the problem of the syntactic parsing. The argument presented there concerned the practical experience with our metagrammar and with the sentences from our testbed. In this chapter we would like to provide one more argument – namely the statement that the complexity of parsing significantly grows with the growing magnitude of nonprojectivity. More precisely, we would like to show that there is a substantial difference in the upper estimation of the parsing complexity of a language containing only sentences with a fixed limit of the value of *dNh* and the lower estimation of the parsing complexity for a (potentially infinite) sequence of Czech sentences where the value of dNh increases systematically with each next element of the sequence, and hence dNh exceeds any limit previously given. A thorough discussion of examples supporting the claim that Czech sentences have this property may be found in [Holan et al. 00].

Let us first introduce some important notions, which we are going to need for a more precise complexity estimation. The first important notion is the notion of *D-parsing*. This notion differs in a certain way from the D-analysis (see Definition 5.6). The task of obtaining all dependency trees directly is too complex (the set of trees can be too large), and also algorithmically not too interesting. We will proceed in a way similar to the CF-parsing or the TAG-parsing, cf. [Sikkel, Nijholt 97], where the parsing means obtaining a set of 'items' of a certain form instead of a set of trees. D-parsing means to obtain a set of items of a similar form, which contains the relevant information about the D-analysis of the 'parsed' sentence, but represented in a much more economic way than it is required by the D-analysis.

Before we start describing more exactly what we understand by the notion of *D-parsing*, we must first introduce a data structure suitable for our purposes. For each node $u_i=[a_i,i,v_i,d_i]$ of a D-tree *T* we define $C_i=[i,Cov(u_i,T)]$. As the formula suggests, this pair contains the information about the index and the projection of a particular node in a D-tree *T*. Even though the pair C_i is defined as consisting of two pieces of data, for our purposes it is more appropriate to represent it as a general *n*-tuple of indices $IC_i=[i, i_1, \ldots ,i_{n-1}]$, where the subscript *n* is an odd integer, $n \geq 3$. The index in the first slot then equals the index *i* from the original definition of C_i above and each pair of indices i_k,i_{k+1} (*k* is an odd integer) represents the span (the leftmost and the rightmost index) of one continuous section of $Cov(u_i,T)$. The number of these pairs directly corresponds to the number of continuous sections in the $Cov(u_i,T)$. We denote the $IC_i=[i, i_1, \ldots ,i_{n-1}]$ as an item of *T*, which corresponds to its *i*-th node.

The full set of items for a particular D-tree *T* is an alternative way of representation of the D-tree *T*. We can see that different D-trees from a D-analysis can share some equivalent items. For example, the D-tree T_2 from Fig. 1.2 with the projections $Cov(u_1,T_2)=\{1\}$, $Cov(u_2,T_2)=\{1,2\}$, $Cov(u_3,T_2)=\{3\}$, $Cov(u_4,T_2)=\{1,2,3,4,5\}$ and $Cov(u_5,T_2)=\{1,2,5\}$ can be represented by the following set of items:

$IC_1=[1,1,1]$, $IC_2=[2,1,2]$, $IC_3=[3,3,3]$, $IC_4=[4,1,5]$, $IC_5=[5,1,2,5,5]$

Definition 6.1 (D-parsing of a sentence)
Let *D* be a D-analysis over A^* and $w \in A^*$. We denote as *D-parsing of w* the set of items $DP(w)=\{IC|IC \text{ is an item of a } T \in D(w)\}$.

Obviously, the set *DP(w)* resembles closely the set of items used in the CYK-parsing scheme or Rytter's parsing scheme, cf. [Sikkel, Nijholt 97]. The difference, however, is that we define *DP(w)* statically, going out from the given set of D-trees, assigned to a sentence w. In this manner we can consider this notion as a notion which originates in (algebraic) linguistics.

For the sake of our explanation we need also the notion of complexity of D-parsing.

Definition 6.2 (Complexity of D-parsing and degree of ambiguity of D-analysis)
Let *DP(w)* be a D-parsing of *w*. We denote the number of items of *DP(w)* as the *complexity of DP(w)*.

Similarly, let *D(w)* be a D-analysis of *w*. We denote the number of D-trees of *D(w)* as the *degree of ambiguity of D(w)*.

Observation
Let *DP(w)* be a D-parsing of *w*. The complexity of *DP(w)* can serve as a lower estimation of the time complexity for any sequential procedure computing *DP(w)*

This observation is based on the following consideration. Any sequential procedure computes the items of DP(w) in some sequential order, and needs at least one step for any item. Therefore it uses at least such amount of steps which equals the number of items in DP(w).

In the same way we can obtain a similar observation for the degree of ambiguity of D-analysis.

The number of D-trees assigned to a sentence *w* consisting of *n* words may be quite large, up to n^{n-1}. However, these trees contain in many cases identical subtrees, which can be computed only once. We may therefore reduce the task of creating all possible trees to the task of creating all possible items. Therefore we further consider only the complexity of D-parsing.

For our purpose it is also important that there is a direct correspondence between the value of dNh for a particular D-tree *T* and between the maximal size of items IC_i – if $h=dNh(T)$, then the maximal size of items IC_i equals $2h+3$. It means that for projective D-trees $(h=0)$ the items IC_i will consist of 3 indices, while for the representation of a non-projective D-tree with two holes it is necessary to use the items IC_i with the maximum of 7 indices. The items representing nodes with the value of *dNh* smaller than maximum will contain only the necessary number of indices, not the maximal one. These facts allow for a (rough) upper estimation of parsing complexity of projective sentences and sentences with a limited value of *dNh*. It is equal to the number of all items IC_i that may be created for the input sentence containing *n* words.

For projective sentences of the length *n* we may get maximally n^3 items – all three indices may acquire any integer value from the interval <1;n> (for the upper estimation it is not necessary to take into account the fact that for example the triple [1,2,3] is not a valid item, because the first index of the triple has to be an integer from the interval <2,3>). A similar idea may be applied to sentences with a limited value of *dNh*. Let us suppose that $m=dNh(w)$ for a sentence w. In such a case the items IC_i will contain up to 2m+3 indices and the (rough) upper estimation of the number of items IC_i equals n^{2m+3}. As in the previous case, we do not take into account that some of these items are invalid and that it is therefore possible to make a better upper estimation. Generally it is possible to say that the upper estimation of the parsing complexity of a language *L* with its D-analysis *D,* where *dNh* is limited by a constant, is polynomial. Or, more exactly, that it equals n^{2m+3}, where $m=dNh$.

The other problem (the lower estimation of the parsing complexity of a language L_c, for example Czech, with its D-analysis D_c, with the value of *dNh* greater than any natural number) needs a different approach. The first thing we have to do is to find an appropriate (infinite) sequence of sentences from L_c with increasing values of *dNh* and possibly rapidly increasing values of complexity of D-parsing.

As has been demonstrated in [Holan et al. 00], Czech theoretically allows for an unlimited number of holes. Some examples presented there do have something in common. The sentences with a high score of *dNh* contain at least two richly modified words. The best candidates are verbs, one of them being a finite verb (V_{fin}) and the other being a verb (V_{inf}) in an infinitive form. Let us mark the dependents of V_{fin} as G_i (dependents of the governing verb) and the dependents of V_{inf} as D_j (dependents of the dependent – subordinated – verb), where integers *i,j* represent the left–to–right order among the dependents of the respective governor. According to the examples presented in [Holan et al. 2000], the sentential pattern of the sentence with $dNh=\left\lfloor \frac{n-1}{2} \right\rfloor$ is then as follows:

$D_1\ G_1 \ldots D_k\ G_k D_{k+1}\ V_{fin}\ V_{inf}$

where *2k+3=n* and the *Cov(V_{inf})* contains k+1= $\left\lfloor \frac{n-1}{2} \right\rfloor$ holes. (For an odd *n* we get exactly $\frac{n-1}{2}$ holes, while for an even *n* we get $\frac{n-2}{2}$ holes.). Fig. 6.1 shows a sketch of a D-tree for this sentential pattern.

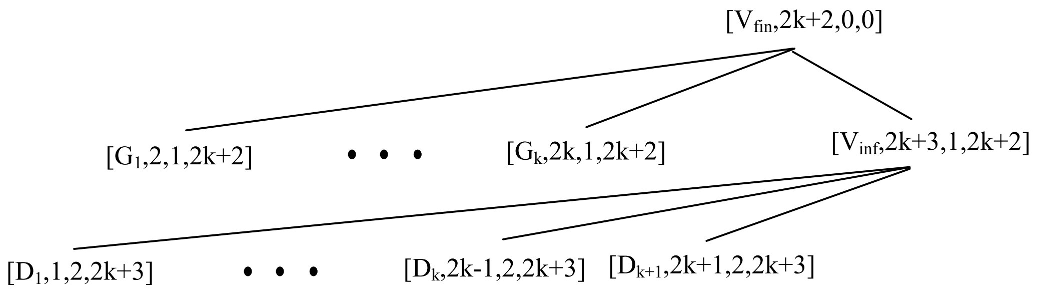

Fig. 6.1. A scheme of a typical nonprojective *D-tree* with *dNg*=k+1

In order to find a Czech sentence with the pattern described above, let us use the following example :

Example 6.1.

Co se komu Petr zítra rozhodl darovat?
[Lit.:What Refl. to_whom Petr tomorrow decided to_give]
(What did Petr decide to give to whom tomorrow?)
Fig. 6.2 shows the D-tree with maximal number of holes for this sentence.

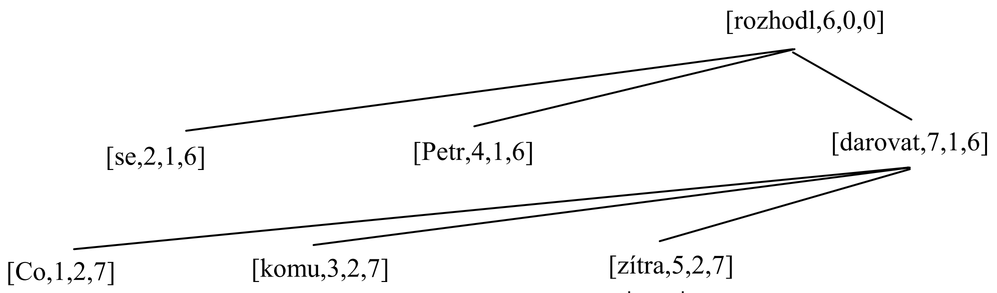

Fig. 6.2. A *D-tree* of the sample sentence with *dNh*= $\left\lfloor \frac{n-1}{2} \right\rfloor$= 3

It is, of course, quite clear that if we try to increase the length of this sentence, we would sooner or later run out of nonprepositional dependents. We would have to use free modifiers in the form of prepositional cases and thus to increase the number of words necessary for creating a hole in the sentence. It means that the lower estimation of *dNh(w)* in the (stepwise increasing) sentence w (of the mentioned type) with respect to its length can be taken as equal to n/4 because of the fact that both the holes and the intervening sections of a projection typically consist of a preposition followed by a noun. Thus the (maximal) length of the strings separating holes is equal to 2 in this case. In the remaining text we will stick to this result.

The fact that a large number of holes implies the presence of a large number of free modifiers[3] in the sentence (we speak about Czech) has a very important consequence. The layout of the D-tree (more specifically, the existence of edges between two particular nodes) depends very much on the interpretation of a particular sentence. This interpretation is, of course, to a large extent dependent on the real-world knowledge of a particular human reader and thus cannot be taken into account during parsing. A syntactic parser should do the opposite – it should provide (in some way) all syntactically plausible D-trees for a given sentence, regardless whether they represent readings acceptable for a

[3] If we used only participants (i.e. subcategorized-for elements) for the purpose, the number *dNh(w)* would indeed have an upper bound, set by the largest number of participants for a lexical head in the language

human or not. The task of the syntactic parser is to analyze the sentence syntactically, not to interpret it.

Let us illustrate this idea by the following sentence:

Example 6.2

Proti odvolání se zítra Petr v práci nakonec důrazně rozhodl protestovat.

[Lit.: Against dismissal Refl. tomorrow Petr at work finally vigorously decided to_protest]

(Petr finally decided to protest vigorously against the dismissal at work tomorrow.)

Figure 6.3 shows the D-tree with a maximal value of dNh for this sentence:

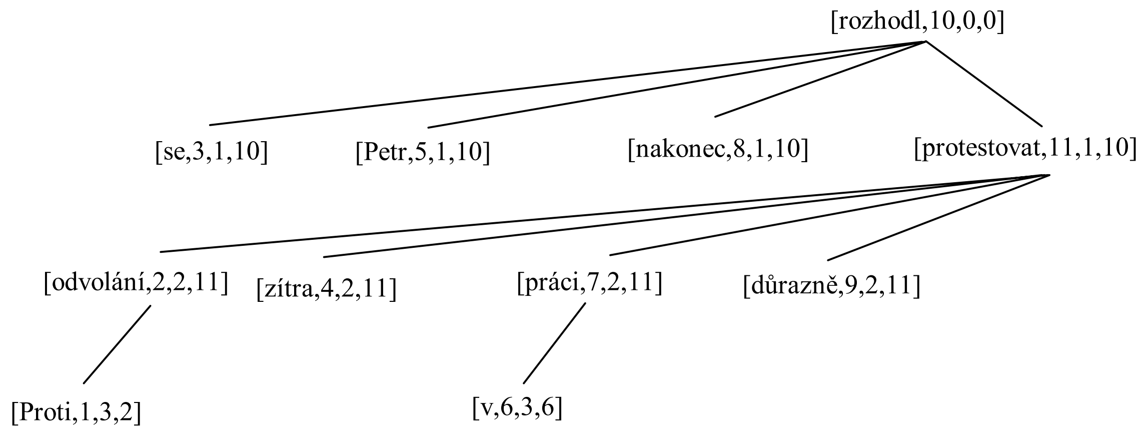

Fig. 6.3. A D-tree representing a maximal possible value of dNh=4 for the sentence *Proti odvolání se zítra Petr v práci nakonec důrazně rozhodl protestovat.*

In order to illustrate the problem with the attachment of free-modifiers we may draw other D-trees representing acceptable results of syntactic parsing of the same sentence.

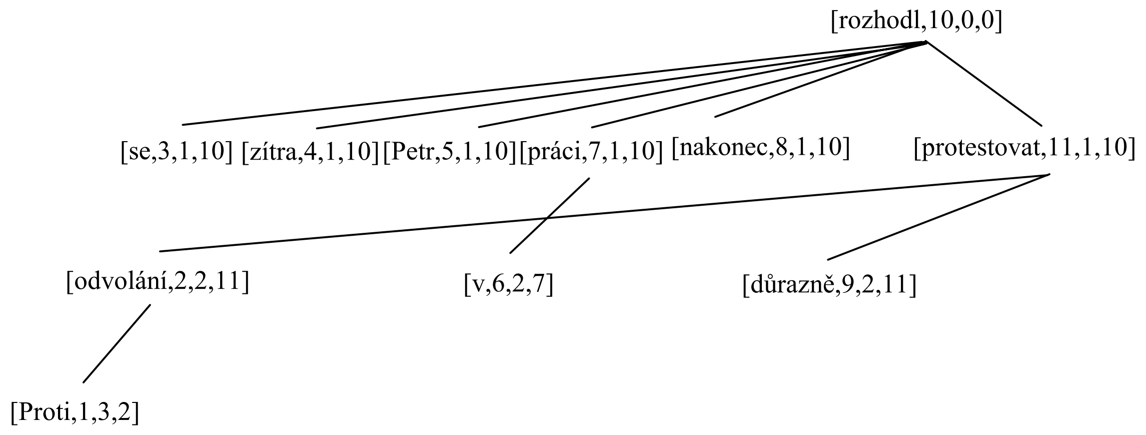

Fig. 6.4. A *D-tree* of the sentence from the example 6.2 with *dNh*= 2

It may be argued that the attachment of the temporal adverbial expressing future (*zítra* – tomorrow) to the verb in past tense in the D-tree from the Fig 6.4 is incorrect. The unacceptability of this reading is not based on the knowledge of context or on the real-world knowledge of the reader, it is based on the lexical semantics of the adverb conflicting with the tense of the verb. Moreover, if we take the word *včera* – yesterday instead of the word *zítra* – tomorrow, the D-tree from the Fig. 6.4 and 6.5 would be semantically correct as well.

However, from the point of view of the "pure" syntax the reading is as plausible as the previous one and the syntactic parser should provide it together with other syntactically correct results.

The parser should provide even more "strange" results, as for example a projective D-tree described in Fig. 6.5. Also this D-tree represents a syntactically acceptable result of parsing:

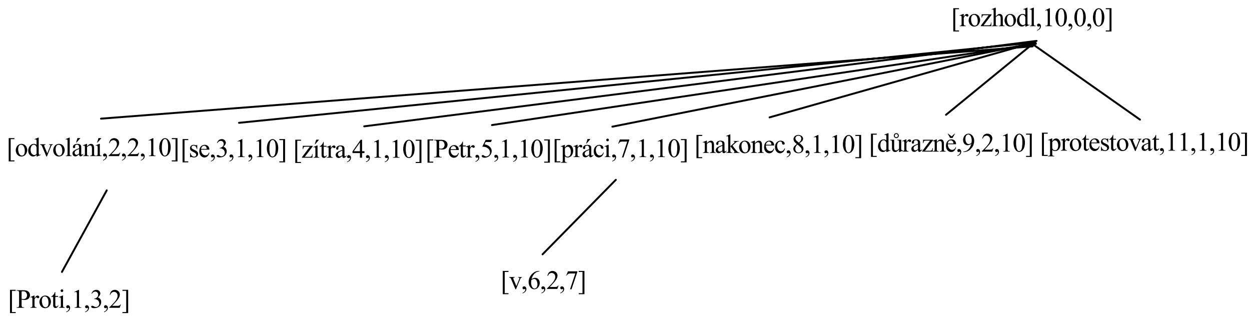

Fig. 6.5. A *D-tree* of a sample sentence with *dNh*= 0

The previous set of examples shows that in a language such as Czech, which allows for an unlimited value of *dNh*, it is necessary to take into account that for any natural number *n* there exists a sentence of length *n* containing *n/4* holes. Moreover, the same sentence may be understood by a human as a sentence containing any smaller number of holes or it may even be understood as a projective one. This claim is based on the fact that when interpreting a sentence containing several free modifiers, the human reader has to rely on extrasyntactic clues (semantics, pragmatics, real world knowledge, context etc.). This may lead to a different interpretation (and thus also to a different number of holes) than the one intended by the author. The task of the syntactic parser (with the stress on the word "syntactic") is to provide information about all syntactically plausible D-trees. Someone or something else, be it a human or an expert system, will then decide which of those D-trees represent the most appropriate reading with respect to the given moment, the background of the reader, the context and other important factors. A very good example of such decisions represents the work on the Prague Dependency Treebank [Bémová et al. 97], where the human annotators are given sentences in their context and they try to create analytical trees reflecting the single most acceptable reading from their point of view even for sentences which are clearly syntactically (or even semantically) ambiguous.

In order to fulfill its task the syntactic parser should compute not only the items with maximal value for a given sentence, but also all syntactically plausible items corresponding to lower values of *dNh*. The number of possible combinations equals $\sum_{i=0}^{m}\binom{m}{i}=2^m$, where $m = dNh$. Given our earlier assumption that for any *n* understood as the number of words in a sentence there exists in Czech at least one sentence which can be considered at the same time as containing *n/4* holes, and as containing *(n/4)–1* holes, ... , and as containing no hole, we have obtained a result we were looking for – the lower estimation of parsing complexity with respect to the length of the input sentence is exponential or, more exactly, it equals $O(2^{\frac{n}{4}})$.

This result supports our earlier claim that the complexity of parsing substantially grows with the growing magnitude of nonprojectivity (taking into account the number of items which have to be derived in order to parse a sentence with a certain magnitude of nonprojectivity). It also justifies our decision to limit the magnitude of nonprojectivity to 1 for the development, testing and debugging of the metagrammar used in our system – due to the fact that identical metarules are used for the parsing of nonprojective sentences regardless whether the sentence contains one or more holes.

At this point we feel the necessity to stress once more the difference between syntactic parsing and grammar checking. As we have demonstrated in this chapter, the task of the syntactic parsing is to provide all syntactically acceptable trees. It must consider non-projective trees wherever possible, even when the sentence is perfectly projective for a human reader. This fact leads to the complexity mentioned above. On the other hand, a grammar checker is concerned with syntactic correctness, it is not concerned with the nonprojectivity as much as a syntactic parser. The main goal of a grammar checker is to find at least one syntactic tree for each input sentence. It is therefore quite natural to orient the grammar checker towards the simplest possible tree – it should look for a projective tree without syntactic inconsistencies first. Only in case such a tree does not exist the grammar checker should try to look either for a non-projective tree with the smallest possible number of holes or it should look for a tree with edges representing syntactic inconsistencies. This strategy is the reason why it is possible to consider grammar checking as a less complex task than syntactic parsing.

To conclude this chapter we would like to remind that the majority of examples used throughout the chapter and also in the paper containing a thorough discussion of the problem of a nonexistence of the upper estimation of the number of holes in Czech [Holan et al. 00] uses simple sentences without any kind of coordination. This fact means that our lower estimation of parsing complexity is very conservative – for complex sentences containing coordination it would probably be possible to obtain a much higher lower estimation.

Chapter 7: Software environment

The actual implementation of the software environment necessary for the development, testing and debugging of the metagrammar was carried out by Tomáš Holan in the LATESLAV project. It is described for example in [Kuboň et al. 97]. We stick to that original version even though the same author has made several newer versions since then. The current state of the implementation is described in [Holan 01]. The implementation was to a large extent influenced by the demand for effectiveness of the whole system. Even though the speed and efficiency are not among the leading criteria for experimental robust natural parser systems, this implementation proved to be suitable also for our purpose.

The implemented formalism requires the use of a very simple data structure representing individual items. The simplicity of the data structure allows faster and more effective interpretation of the grammar. The data structure is a set of attribute-value pairs with the data about valency frames of particular words as the only complex values (embedded attribute-value pairs). The description of data structures for particular word categories can be found in Chapter 8.

As was already briefly mentioned in the previous chapter, it would be very difficult to use directly the rules of RFODG in the attempt to develop a formal grammar of a natural language. With respect to the number of terminal symbols (representing the morpho-syntactic and lexico-syntactic properties of individual word forms) used by the grammar it is virtually impossible to write grammar rules covering a substantial part of any language. For this reason we use a kind of a two-level description of a grammar by means of metarules, each of which represents a finite (but typically very large) number of rules of RFODG. The metagrammar is then a set of metarules accompanied by constraints. The metarules are interpreted in a manner given by the set of constraints used.

The metarules have a common general form A B $\Rightarrow$ X where the letters A and B represent already existing items (from the point of view of an application of an individual metarule we sometimes refer to these items as to the input items of a metarule). The item A always stands to the left from the item B (each item represents a particular word form and has a horizontal index referring to the position of the word in the input). In case a particular metarule may be applied to items A and B, a new item X is created (X represents an output item from the point of view of the application of an individual metarule to a given pair of input items A and B). The metarules express a procedural description of the process of checking the applicability of a given metarule to a particular pair of items A and B. In addition to the items A, B and X the metarules use also the temporary item P. The commands and constraints of individual metarules are interpreted from the beginning sequentially with respect to their order.

The syntax of metarules does not allow to create metarules with a reduced left-hand side either of the type A => X or B => X. This means that the type of rules mentioned in the definition of RFODG under b) (A->B, where A, B $\in$ V) were not implemented in this version of our formalism.

Every metarule in our metagrammar typically starts with a header, consisting of comments containing the rule number and a short description of the objective of the metarule. This part is not obligatory, but it serves for a better orientation in the grammar during its development, testing and debugging. The interpreter numbers the metarules sequentially, starting with the rule number 1. These numbers appear (when required, their appearance is triggered by a parameter) also in the graphs representing results of application of a metagrammar to a sentence. It is therefore a good idea to use the same kind of numbering. It may of course be argued that the interpreter could number the metarules automatically. At this point it is necessary to stress that the software environment was developed primarily as a development tool, it was being developed together with the earlier versions of the metagrammar and thus it underwent a number of changes. It is then no wonder that the interpreter lacks to a certain extent the user-friendliness.

The header may be followed by the following parameters:

7.1 Parameters of metarules

Any combination (even the empty one) of the three parameters **PROJECTIVE**, **CLOSEST** and **NEGATIVE** may precede all commands and constraints in a particular metarule.

PROJECTIVE means that the metarule may be applied only in a projective way – it is in principle the same as a rule of a standard CFG. In the definition of the RFODG this parameter represents the values of **X**$\in$ **{LP,RP}.** This of course does not mean that the application of the rule with this parameter should be restricted to projective phases – it only means that regardless of the type of the phase this particular rule may be applied in a projective way only. If this parameter is omitted, then it means that in projective phases the metarule in fact represents the rules of RFODG with the values of **X**$\in$ **{LP,RP}**, while in nonprojective phases the value of X $\in$ **{LP,RP,L,P}**.

NEGATIVE means that the metarule is taken into account only in an extended grammar (in the grammar working both with positive and negative symbols). The metarules of this type are in fact error anticipation rules – they describe typical syntactic inconsistencies that are frequent in Czech texts.

CLOSEST indicates that the metarule may be applied only to the closest input items A and B, in the sense of a static order of input words. It means that if the input items A' A" B' B" are in this order, the metarule may be applied only to the pair of items A" B'.

The header is followed by a body of a metarule. It consists of a sequence of instructions and constraints, using the following elementary instructions:

Hard constraint

A.x = const

or

A.x = B.y

where A and B represent input items, x, y are their attributes and "const" is a constant of the same type as the attribute A.x.

The interpretation: If the equation does not hold, then this rule cannot be applied for particular input items A and B. This type of constraints is interpreted in the same way in both the positive or negative parsing phases.

Soft constraint

A.x ? const   ERR

or

A.x ? B.y   ERR

where A,B,x,y and "const" are the same symbols as in the previous instruction and ERR is a code of a syntactic inconsistency.

The interpretation: The soft constraints may be relaxed in negative phases. In such a case the soft constraint does not block an application of the metarule even in case when the constraint is violated. The code of the encountered syntactic inconsistency (ERR) is written into the output item in such a case. In positive phases they are applied in the same manner as the hard constraints. These constraints typically describe errors in agreement, but they are able to cover much wider range of errors. Several soft constraints may be relaxed at the same time.

Assignment

X.x := const

or

X.x := A.y

where X is an output item, A is an input item, x, y are their attributes (of the same type) and "const" is a constant of the same type as the attribute X.x.

The interpretation: If the attribute x is not present in the data structure X, it is added. The copy of the value of the attribute A.y becomes the value of X.x. This instruction allows to insert into the resulting item an important information which may later be propagated through relevant items. The previously non-existent attributes inserted by means of the assignment are most often used for transport of important information bottom up through the tree. They may, for example, contain the information about the type of items which were already combined with the item containing this attribute – cf. the attributes *dep_neg, dep_rel, dep_dem* etc. in the comments accompanying the metarule (1) in Chapter 9. In the following text we call this type of attributes *auxiliary attributes*.

Branching

IF <condition> **THEN** <instructions1> **ELSE** <instructions2> **ENDIF**

The interpretation: When the <condition> following the keyword IF is not met, the computation does not fail but continues with <instructions2> or after ENDIF if ELSE is missing. If the <condition> is met, the computation continues with <instructions1> following the keyword THEN. Nested branching is allowed.

The choice of an element from a set

P in A.x

where P is a temporary and A an input item, x is an attribute of A.

The interpretation: If the item A has no attribute x or if the value of this attribute is not a list or if the list is empty, then the application of the rule fails for the data it is being applied to.

In the opposite case the current position in the program is stored, P acquires the value of one element from the list A.x and the computation continues.

When the computation is finished, no matter whether successfully or not, the program returns to this point and tries to proceed taking another item from the list A.x, until all items were tried.

This command may create more output items from one input item.

Deleting in a list

\ P from X.x

where P is a temporary item, X.x is an attribute whose value is a list containing the same value as P.

The interpretation: The item equivalent to P is deleted from the list of values of the attribute x (if x contains a list of values). If the deletion cannot be performed, the computation fails.

Comment

;

The semicolon starts a comment, which automatically ends at the end of the same line

There are two keywords, which may be used throughout the body of the metarule. These keywords end the interpretation of the metarule. There may be several keywords of this type in one metarule:

Success

OK

The interpretation: Success. The output variable X contains an item, which is the result of the application of the rule on the input items A and B. This keyword must be preceded by an assignment of one of the items A or B to X (X:=A or X:=B). This assignment determines which of the two input items is the governing one. Such assignment means that all attributes and values of the input item A (or B) are copied into the output item X.

Failure

FAIL

The interpretation: Failure. The rule is not applied to the input items. No new item is created. If the interpreter encounters the keyword FAIL, the interpretation of the metarule stops immediately.

Every metarule ends with a special keyword. This marker serves for the orientation only, in fact it is never reached during the intrpretation of the metarule because one of the special codes FAIL or OK must be reached earlier.

End of a metarule

END_P

This keyword marks the end of a metarule.

Example 7.1

An example of a (simplified) metarule describing the attachment of a nominal modifier in the genitive case from the right hand side of the noun (e.g. "ministr[nom.] dopravy[gen.] " – minister of transport). For a better readability of the metarule the comments are included in brackets {} instead of being preceded by a semicolon:

```
PROJECTIVE
CLOSEST
        {This rule may be applied only in a projective construction and only to A and B which are
        immediate neighbors.}
IF A.SYNTCL = noun THEN ELSE
        IF A.SYNTCL = prephr THEN ELSE FAIL ENDIF
ENDIF
        {If the symbol A represents an item which is a noun or a prepositional group then go on, else
        fail and do not continue with the application of this rule.}
B.SYNTCL = noun
B.CASE = gen
        {The symbol B represents an item, which is a noun in the genitive case}
A.RIGHTGEN ? yes     Second_genitive
        {Soft constraint checking if there already has been a noun in the genitive case attached as a
        right hand side modifier before the application of this rule. This is necessary, because Czech
        does not allow a noun to be modified by more than one noun in genitive case from the right
        hand side. It might be argued that in case of coordination of two or more nouns in the genitive
        case it would be syntactically correct to attach more than one noun in the genitive case. It is
        generally true, but our metagrammar does not allow it. It always tries to process the
        coordination first and to attach it as one item (subtree). The detailed description of metarules
        of our metagrammar is contained in Chapter 9.
        The string " Second_genitive " is the code of a syntactic inconsistency, which is used in case
        the soft constraint is violated in the negative phases.}
X:=A
        {The symbol A is copied into the newly created item X.}
X.RIGHTGEN := no
        {The value of an attribute RIGHTGEN of the item X is modified in order to block the
        attachment of the second noun modifying the same item A}
OK
        {The keyword OK confirms a successful application of the rule on the given pair of symbols
        A and B.}
```

END_P
{Keyword marking the end of the rule.}

7.2. The interpretation

The interpretation of the metagrammar is performed by means of a slightly modified CYK algorithm (a description of this algorithm may be found among other books also in [Sikkel 97]). The grammar works with unambiguous input data (ambiguous words are represented as sets of unambiguous input items with the same horizontal index – cf. the definition of DR-trees in Chapter 5). The interpreter also takes care of the duplicity of certain syntactic structures (subtrees) during the process of parsing. In nondeterministic parsing it is often the case that we might get the same subtree by an application of certain rules in a different order. This situation was quite common in the Czech parser of the system RUSLAN due to the fact that Q-systems work in a nondeterministic manner. As an example of such a situation we may take the sentence *Karel poslal dopis* (Charles sent [a] letter). If the parser attaches object to the verb first, we get the situation from Fig.7.1:

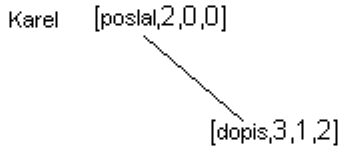

Fig. 7.1. An object attached to the verb in the sentence “Charles sent a letter”

If the rule for the attachment of the subject is applied first, the parser will create the structure from Fig.7.2:

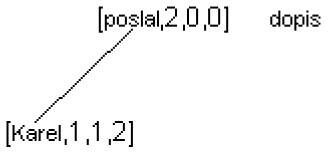

Fig. 7.2. A subject attached to the verb in the sentence "Charles sent a letter"

A nondeterministic parser will create both structures. The main problem is that later it will attach the subject to the verb from Fig.5a and the object to the verb from Fig.5b. As a result of these attachments it will in both cases create the identical structure from Fig. 6:

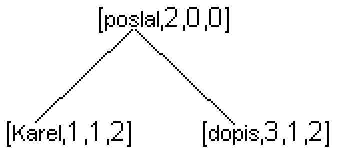

Fig. 7.3. The syntactic structure of the sentence "Charles sent a letter"

The existence of such spurious ambiguities is a great pain, they can substantially slow down the parsing or even block the success of the parsing by exhausting available resources (for example the memory). One solution to this problem was implemented in the system RUSLAN, where the left-hand side attachments were preferred to the right-hand side ones. Such approach may cause more problems rather than it solves, since sometimes it might be difficult to find out whether all left-hand side modifiers of a particular word have already been attached. The other important problem is that in case of a failure of an attachment of some of the left-hand side modifiers the parser might not be able to continue in attaching the right-hand side ones. Another argument against the use of such techniques is the fact that there usually is no linguistic reason for such preference of modifiers from either side, especially in languages with a high degree of word order freedom, where many modifiers may appear on both sides of the governing word.

The interpreter of RFODG works in a different manner. It can apply various kinds of global constraints on a set of items created during the interpretation of a particular metagrammar not only at the end of the process, but also during it. There are several kinds of global constraints applied during the parsing process, among others also the constraints on the degree of nonprojectivity which were mentioned in the previous chapter. Spurious ambiguities are handled by a global constraint which filters out all items with the same structure immediately after they are created. In this way the grammar is not burdened with additional commands and conditions which make the parsing process more complex than necessary and the metarules may concentrate on really linguistically motivated problems.

Another global constraint is used in negative phases. This constraint suppresses syntactic inconsistencies wherever there exists a "correct" (i.e. without inconsistency) item. The necessity to apply this global constraint is a consequence of the fact that the input items of the parser are unambiguous in the sense that each morphologically or syntactically ambiguous word is represented by a set of unambiguous input symbols. For example, if a preposition may be associated with a noun either in the accusative or in the locative case, then it is represented by two input items, one representing the preposition "governing" accusative case, the other representing the same preposition "governing" the locative case. This general property of all input items causes a substantial problem in negative phases.

For example, a typical error in free word order languages is an agreement error. Let us suppose that we have the following three input words as immediate neighbors in the order presented below:

1. **Preposition** ("governing" accusative or locative case) – [2 unambiguous items]
2. **Adjective** (masc. inanimate gender, nominative, accusative or vocative case, masc. animate gender, nominative or vocative case, singular) – [5 unambiguous items]
3. **Noun** (animate gender, genitive or accusative case singular) – [2 unambiguous items]

These words are represented by nine unambiguous items. If we try to create a nominal group in prepositional case without the constraint relaxation, we get one resulting item representing the nominal group in an animate gender, accusative case and singular number. On the other hand, after the relaxation of constraints there are 20 items created. One of them (the one mentioned above) does not contain any syntactic inconsistency, 2 items contain one syntactic inconsistency and the remaining 17 have two. Without the application of global constraints filtering out all items with at least one syntactic inconsistency all 20 variants would be used in the subsequent parsing. This would cause a combinatorial explosion of mostly irrelevant results.

The global constraint works in the following manner: Every time an item representing a new branch or subtree is created, it is compared with the items representing other branches or subtrees with the same structure and projection. If it is "worse" than those already existing according to the measures applied (if it has a higher degree of robustness), it is not parsed further.

This technique substantially reduces the number of items in negative phases, but it has also one rather unpleasant side effect: the syntactic inconsistencies may be suppressed and appear later in a different position in the tree.

Let us demonstrate this undesirable property by the following example. Let us suppose that we parse the sentence:

Začátek novému roku nám přináší novou naději.

[Lit.: Start new[...,nom.sg.,...] year[gen.sg,dat.sg.,loc.sg.] to_us brings new hope.]

(The start to the new year brings a new hope to us.)

The problem of this sentence is the nominal group *začátek novému roku*, which contains a syntactic inconsistency (the adjective has a dative instead of the correct genitive form). The nominal group locally does not contain any inconsistency, since it may be parsed as being in the dative case. Our interpreter will never find this type of inconsistency. It will locally parse the nominal group as one of the three candidates for filling the valency slot of the main verb. Only later, when trying to fill the two valency slots of the verb with three candidates, it will report a syntactic inconsistency of a completely different type. The superfluous nominal group in the dative case may neither have a role of a free modifier in the dative case in this sentence due to the fact that there is already one nominal group in the dative case present (*nám* – to_us) in the sentence. The problem related to the difference between syntactic inconsistencies and errors is not a substantial one, because a simple module comparing and evaluating syntactic inconsistencies of all syntactic trees obtained as a result of robust parsing of a particular input sentence may take care of such cases.

Chapter 8: Syntactic dictionary

In every natural language parsing system it is necessary to make one very important decision – to which extent the syntactic information is going to be divided between the (meta)rules of a (meta)grammar of the system and the syntactic dictionary of the system. It is clear that any extreme solution (either all relevant information is present in the grammar and the dictionary is almost empty or the majority of syntactic phenomena is encoded in the dictionary and the grammar contains only a few basic rules) requires more effort than the system in which the syntactic information is balanced between the dictionary and grammar. During the last years there was, however, a general trend towards the latter solution. It has several advantages for small–scale experimental systems, but for a dictionary of hundreds of thousands of lemmas it is very difficult to build and maintain a consistent syntactic dictionary.

Also in the previous attempts to implement a syntactic dictionary for Czech it was always the case that the division of the information was slightly in favor of the dictionary. The main reason is probably the fact that in languages with relatively free word order the valency information is substantial for building a syntactic representation of sentences and thus the dictionary information is very important. This claim is also supported by the fact that Czech verbs may have a wide variety of combinations of participants in their valency frames (cf. [Panevová 80]). It is very difficult, to categorize the verbs into classes with identical valency frames, there are simply too many variants (even though the most typical combination [nominative – accusative] in the active mode and [nominative – instrumental] in the passive mode is quite frequent). For this reason it is better when the valency information is attached to individual words in the dictionary.

The basic idea of the split of syntactic information in our robust parser is quite simple – basic properties of all well–defined groups of words should be written in the form of grammar metarules, the specific behavior of individual words should be contained in the dictionary. This means, for example, that the fact that every noun may have a postponed nominal modifier in genitive and that this modifier can be only one, is a part of the metagrammar, while the information that the verb *žádat* [to ask] is not reflexive belongs into the syntactic dictionary.

8.1 Information contained in the dictionary

As we have already mentioned in previous chapters, the data are stored in the form of a set of attributes and their values. The actual nature of data stored in the syntactic dictionary of the system varies to a great extent with respect to the particular part of speech. There are nevertheless some attributes that are common (we may even say obligatory) for all types of words:

lexf is the basic form of the word, which identifies the word in the dictionary and allows a mapping with the morphemic information acquired by the morphological analysis;

wcl represents the “morphological” information about the part of speech of the word. This information corresponds to the part of speech information contained in lexicons written for human readers;

syntcl represents the "syntactic" part of speech of the word. The value of this attribute represents the syntactic behavior of the word. We make distinction between *wcl* and *syntcl* for the reason that in many cases the information about the part of speech (*wcl*) does not give a sufficient clue how to handle a particular word during syntactic analysis – for example pronouns are very often treated similarly to adjectives or nouns. In other words, a word with the value of *wcl* equal to *prn* very often has a *syntcl* with the value either *adj* or *noun* with respect to the part of speech of words, which it syntactically resembles.

There is one more attribute which is common for several categories of words – the attribute *frameset* contains the information about the valency frame of the given word (at this point it is necessary to stress that it contains only the information relevant for our purposes, not the information which belongs to the valency frame from the point of view of linguistic theory). This is the only attribute, which may have complex values.

The syntax of this attribute is:

frameset: ? ([slot]...[slot]) , ... , ([slot]...[slot]) !

where each *slot* is a triple of attributes

actant contains the name of the participant represented by this slot

case stores the information about the case of the nominal participants or about the type of the participant in case the participant is not nominal (e.g. infinitive or clause)

prep represents a conjunction in case the participant is a clause (in our simplified valency frames no prepositional groups are present – cf. the discussion in section 8.4).

The symbol *?* marks the start of the list of frames, commas delimit individual frames and *!* marks the end of the list. These symbols are used generally in all dictionary items as markers for variants both at the level of values and at the level of attributes. It is possible to embed more levels of these markers one into the other. In such a case they are not allowed to cross, the innermost question mark always matches the innermost exclamation mark.

The structure of individual classes of words resembles the structure of the syntactic dictionary used in the LATESLAV project [Skoumalová 94]. Since the dictionary in that project was supposed to be a general resource for different variants of grammar checkers (there were in fact three versions of the Czech grammar checker developed in the frame of the project [Kirschner 94, Oliva 96, Kuboň et al 97]), it contains a lot of information irrelevant for the purpose of robust syntactic parsing (for example the description of the meaning of homonyms, semantic features etc.). For this reason it was necessary to modify it in the way described in the following paragraphs.

8.2 The structure of individual classes of words

In the sequel, the three obligatory attributes *lexf, wcl* and *syntcl* are not listed. On the other hand, not all attributes listed bellow are explicitly a part of the description in the syntactic dictionary. If, for example, a verb is not reflexive, it does not have the attribute *refl* at all. Some attributes are also present only virtually and appear in the structure only when they are needed during the parsing process. This is, for example, the case of the attribute *rightgen,* which is inserted into the structure with the value equal to *no* in case there is a nominal modifier in genitive attached to a particular noun from the right hand side. These attributes are more or less auxiliary and are listed separately. The sole reason for the split of attributes into groups of syntactic and morphemic ones below is the readability of the list of attributes, it has no function neither in the dictionary nor in the metagrammar.

8.2.1 Nouns

Syntactic attributes:

tant marks if the particular word is a singulare (*sg*) or plurale (*pl*) tantum or none of these (*0*)

refl information about the required type of reflexive particle (*se*, *si* or none (*0)*)

frameset contains the information about the valency frame

title is an attribute marking the possibility of using a particular word as a title (cf. *comrade* Zeman or *chairman* Mao). It is clear that the number of nouns which may be used in the syntactic role of a title is rather big (cf. that stupid *bitch* Vlasta, the bloody *murderer* O.J. etc.), but only those used regularly and often in this role get this attribute.

In the process of parsing the following **morphemic attributes** are also used:

gender marks the gender (*anim*ate, *inan*imate, *fem*inine or *neut*ral) *num* contains the information about number (*sg* or *pl*)

du informs whether the word form is dual or not

case contains the information about the case of the word form. Czech has seven cases (*nom*inative, *gen*itive, *dat*ive, *acc*usative, *voca*tive, *loc*al and *ins*trumental).

Among the **auxiliary attributes**, which might be used during the parsing process and are not explicitly listed in the dictionary, there are the following ones:

rightgen – equals *no* if a nominal modifier in the genitive case is already attached to the given noun from the right hand side

dep_poss, dep_pers, dep_refl, dep_dem, dep_que, dep_indef, dep_neg, dep_rel, dep_tot are the attributes which indicate that a certain type of a pronoun (the names of these attributes correspond to the values of the attribute prn_type, cf. chapter 8.2.3) already depends on the noun from the left–hand side

dep_card, dep_ord, dep_sort, dep_spec indicate that a particular type of a numeral (the names of these attributes correspond to the values of the attribute num_type, cf. chapter 8.2.4) already depends on the noun from the left–hand side

prep contains the lexical value of a preposition in case the noun is in a prepositional case

8.2.2 Adjectives

Syntactic attributes:

adj_type is either *ord*inary, *aposs* (possesive), *vact* (verbal active) or *vpass* (verbal passive form)

deg represents the degree of an adjective (*pos*itive, *comp*arative or *sup*erlative)

neg negation (*yes* or *no*)

frameset contains information about the valency frame

refl information about the required type of the reflexive particle (*se*, *si* or none (*0)*)

Morphemic attributes:

nomform short (nominal) form of an adjective (*yes* or *no*)

gender, *num*, *du*, *case* are same as the attributes of nouns with identical names

8.2.3 Pronouns

Syntactic attributes:

prn_type represents a type of a pronoun – *pers*onal, *refl*exive, *rel*ative, *dem*onstrative, *indef*inite, *neg*ative, *poss*esive, *que* (interrogative) pronouns and *tot*alizers

encl – clitic *yes* or *no*

pprep informs whether a preposition is required in front of the form of the pronoun (cf. *ji* but pro *ni* [her – for her])

Morphemic attributes:

gender, *num*, *du*, *case* are the same as the attributes of nouns with identical names. Those pronouns, which do not have these categories, also do not have these attributes (the attributes are not obligatory).

pers means person (1st, 2nd or 3rd)

8.2.4 Numerals

Syntactic attributes:

numcl represents a class of a numeral (*card*inalia, *ord*inalia, *spec*ialia, *mult*iplicativa and generic (*ord*))

numtype is a type of a numeral (*rel*ative, *que* (interrogative), *dem*onstrative, *indef*inite, *tot*al, *dig*it)

Morphemic attributes:

gender, *num*, *du*, *case* are the same as the attributes of nouns with identical names. Those numerals, which do not have these categories, also do not have these attributes (the attributes are not obligatory).

8.2.5 Verbs

Syntactic attributes:

v_cl is a class of verb – *full*(main), *aux*iliary

refl contains information about the required type of reflexive particle (*se*, *si* or none (*0)*)

aspect informs about the aspect of the verb (*prf*(perfective), *impf*(imperfective) or *iter*ative)

v_form is a form of the verb – *fin*ite, *inf*inite, *past p*articiple, *passp* (passive participle) or *transgr*essive

frameset contains information about the valency frame

neg negation (*yes* or *no*)

encl – clitic *yes* or *no*

Morphemic attributes:

tense is either *past*, *pres*ent or *fut*ure

pers is a person

num contains information about number (*sg* or *pl*)

gender marks the gender (*anim*ate, *inan*imate, *fem*inine or *neut*ral)

pass refers to the ability to form the passive voice: *per*iphrastic, *rfl* (reflexive) or none (*0*)

mode is *ind*icative, *cond*itional or *imp*erative

Auxiliary attributes

clause indicates the type of the clause governed by the verb

relgender, relnum contain information about the gender and number of the antecedent of a relative clause. This way of the transfer the gender and number of relative pronouns is necessary due to the fact that we have decided to attach relative pronouns to main verbs of relative clauses and thus there are no edges leading directly from the relative pronoun to its antecedent in our syntactic trees (cf. section 9.6 decribing the metarules handling the attachment of relative clauses).

8.2.6 Adverbs

Syntactic attributes:

adtype is a type of an adverb – *rel*ative, *que* (interogative), *dem*onstrative, *indef*inite, *neg*ative, *tot*al, *adl* (main)

addep is the part of speech of the word the adverb depends on (from the left or right hand side) – *advb* (verb), *add* (adverb), *ada* (adjective), *adnr* (noun).

frameset contains the information about the valency frame

encl – clitic *yes* or *no*

deg represents the degree of the adverb (*pos*itive, *comp*arative or *sup*erlative)

neg negation (*yes* or *no*)

8.2.7 Prepositions

Syntactic attribute:

case is the case of the noun to which the preposition belongs (typically 2 to 3 cases).

8.2.8 Conjunctions

Syntactic attributes:

conjtype is a type of conjunction, either *coord*inate or *sub*ordinate

conjcl specifies, whether the conjunction connects clauses (*cls*) or *phr*ases

compar indicates that the conjunction may be used in comparative constructions

pair indicates that the conjunction may belong to a pair of conjunctions *nejen – ale i* [not only – but also], *ani – ani* [neither – nor] (cf. section 9.12 Pairs of expressions).

8.2.9 Interjections, particles and delimiters (.,:;!? etc.)

They do not have any special attributes except *lexf, wcl* and *syntcl.*

The **auxiliary attributes** *eos, eoc, end* may be attached to all types of governing words. They indicate that the end of the sentence or clause was already processed. The detailed description of their role may be found in section 9.1. among the comments explaining the function of the metarule (3).

8.3 Sample input data structure

The following data structure represents the verb *nepředpokládá* – [{he/she/it does} not suppose] as it appears at the input to the robust parser. It contains the complete set of attributes and their values, with combined morphemic and syntactic data. The first row of data represents the word form as it appeared in the input sentence, the keyword END marks the end of the data structure. Every input sentence is represented by a set of data structures of this type in the order given by the surface order of words in the sentence. The sentence starts with the data structure representing the left sentinel (a special marker determining the beginning of each sentence).

```
nepředpokládá
lexf: předpokládat
wcl: vb
syntcl: v
v_cl: full
refl: 0
aspect: impf
frameset: ? ( [ actant: act
 case: nom
             prep: 0 ]
            [ actant: pat
             case: acc
             prep: 0 ] )
            ,
            ( [ actant: act
              case: nom
              prep: 0 ]
            [ actant: pat
              case: clause
              prep: že ] )
              !
mode: ind
neg: yes
v_form: fin
gender: ? anim , inan , fem , neut !
num: sg
pers: 3
```

tense: pres
END

8.4 Data contained in valency frames

As we have already mentioned in previous chapters, the strategy of our approach to robust parsing reflects to a great extent the fact that a robust syntactic parser should be able to analyze (accept) also syntactically ill–formed sentences. The question of acceptability of a particular input sentence as a well–formed sentence of a given natural language is to a great extent influenced by semantics. The task of drawing the line between semantically plausible and implausible constructions is according to our approach beyond the scope of this thesis. Unlike purely syntactic rules, semantics and pragmatics do not constitute a solid base for an automatic decision about the acceptability of a particular sentence. Semantics, pragmatics and real–world knowledge involved when native speakers decide whether to accept or reject a particular sentence is too complicated to provide a well–defined tool usable in the automatic language processing.

These considerations had a strong direct influence on our work. It was necessary to modify the theory in order to avoid problems that are not solvable by means of purely syntactic tools at the level of surface syntactic parsing. Basic information for building a tree representing the syntactic structure of a sentence is encoded in the valency frames of verbs. Without them, it would be hardly possible to achieve reasonable results in the course of parsing and thus to avoid a flood of syntactic structures as the product of the analysis of any sentence. The valency information allows to attach inner participants to the main verb and thus to build a backbone of the syntactic structure. According to the theory we subscribe to ([Panevová 80], [Sgall et al. 86]), the participants are, with respect to the necessity of their presence, divided into two basic groups, obligatory and optional. This division holds on the level of tectogrammatical representation of the sentence. On the surface level we have to take into account that in the textual context it is often the case that one or more obligatory inner participants are explicitly expressed in the previous context and can be omitted in the surface form of the current sentence. Especially in dialogues it is acceptable to use sentences like: *Poslal Karel Mileně včera ten dopis? Poslal.* [Did Karel send that letter to Milena yesterday? lit.:Sent.]

For this reason it has no sense to check whether all valency slots are filled or not and to deal with an input sentence as if it were ill–formed in case some of the obligatory participant is missing. That also means, on the other hand, that even certain kinds of sentences considered by human native speakers as clearly ill–formed are going to be accepted by our parser as well–formed. This is an unavoidable consequence of the strategy chosen for the robust parser.

On the basis of syntactic rules we are also not able to distinguish between participants and free modifiers in general, and also between obligatory and optional inner participants. The reason is that in most cases the distinction between an inner participant and a free modifier is based solely on extrasyntactic factors. Let us illustrate these facts by the following examples:

Shodli jsme se jen na jediném řešení[loc.].
[Agreed [we] ourselves only on one solution[loc.].]
(We agreed on a single solution only.)

Shodli jsme se jen na chvíli[acc.].
[Agreed [we] ourselves only for [a] while[acc.].]
(We agreed only for a while.)

**Shodli jsme se jen na sestru*[acc.].
[Agreed [we] ourselves only for sister[acc.].]
(We agreed only on a sister.)

The asterisk in front of the third sentence means that a native speaker will probably mark this sentence as unacceptable. From our point of view the third sentence is acceptable, because it cannot be distinguished from the second sentence by means of surface syntax only. Slightly different is the situation with nonprepositional inner participants.

Karel mu[dat.] *představil Milenu*[acc.].
[Karel him[dat.] introduced Milena[acc.].]

(Karel introduced Milena to him.)

Karel ho[acc.] *představil Mileně*[dat.].}
[Karel him[acc.] introduced [to] Milena[acc.].]
(Karel introduced him to Milena.)

**Karel ho*[acc., gen.] *představil Milenu*[acc.].}
[Karel him[acc., gen.] introduced Milena[acc.].]
(Karel introduced him Milena.)

In this case the parser can mark the last sentence as ill–formed with respect to the fact that the sentence either contains two objects in accusative or one of the objects is in a wrong case.

These examples show that the only category of participants which provides a sufficient clue for the rejection or acceptance of certain sentences on the basis of surface syntax is the category of inner participants in nonprepositional cases. We can not check the participants requiring a certain preposition. Since every preposition may serve also as a prepositional case of free modifiers (adjuncts), which are acceptable with the majority of verbs, it is impossible to check incorrect prepositional cases of participants.

Unfortunately, the situation in Czech is not so straightforward. The nouns in prepositional cases are not the only candidates for free modifiers. It is also quite common that a noun in instrumental case is a free modifier (it typically expresses manner), much less frequent is a free modifier in dative case and rare are free modifiers in other cases (for example free modifiers in the accusative case may express temporality and measure). To draw a clear line capturing all subtleties of the problem is really impossible. The classification of participants is too much influenced by semantics that we had to choose a solution that is, according to our opinion, quite simple and works with reasonable accuracy. This solution handles nouns in prepositional cases and nouns in instrumental case as free modifiers while nouns in nonprepositional cases are listed in the valency frames of our dictionary entries wherever they have a role of an inner participant. This is a substantial simplification of the problem, but clearly necessary, if we want to make a clear borderline between syntax and semantics.

The simplification of valency frames has also one pleasant side effect – although it would be almost impossible to automatically establish full valency frames (the major problem being how to distinguish for a particular word whether a nominal group in prepositional case encountered in the sentence belongs to the valency frame or whether it is a free modifier), we can try to identify automatically at least participants for our reduced frame. The method of semiautomatic addition of syntactic data into the dictionary might be an interesting topic of research for the future.

Chapter 9: Implementation of the metagrammar

This chapter contains the description of core elements of the metagrammar of the system. Not all metarules are listed in the following paragraphs – only those which illustrate certain interesting problems and demonstrate the solutions to these problems, as they were implemented in the system. Since the order in which particular metarules are written in the grammar does not matter, it is possible to introduce sample metarules dealing with particular grammatical phenomena in one block. They are numbered differently than in the grammar in order to allow consistent references in this report. The rules are taken directly from the source file PRAVIDLA.DAT (by default, the file containing the current version of the metagrammar has to have this name). They are not simplified. All comments are included in curly brackets {}. The symbol A always represents an item standing to the left from the item B, the item X represents a newly created item (in case a particular rule succeeds). The metarules use the syntax described in Chapter 7.

9.1 A nominal group

The first group of metarules presented here deals with the problem of congruent attributes. There are three metarules in the metagrammar dealing with the attachment of congruent attributes from the left–hand side. One metarule concerns proper adjectives, the other two deal with pronouns and numerals. All three rules were written according to the results of linguistic research contained in [Bémová 96]. Let us introduce for example the metarule dealing with the attachment of pronouns that have syntactic properties of adjectives to the noun governing the phrase.

Metarule (1)

A congruent pronominal attribute

A.syntcl = adj

{The item A represents a word with syntactic properties of an adjective}

IF B.syntcl = noun THEN ELSE

{According to the syntax introduced in Chapter 7 the omission of instructions between THEN and ELSE means that if the condition is met, the processing of the metarule continues. Such a construction is very frequent in our metagrammar.}

```
IF B.syntcl = select THEN ELSE FAIL ENDIF
                    ENDIF
```

{The item B either has syntactic properties of a noun or it represents a head of a selective construction – cf. the section 10 of this chapter.}

X := B

{If the rule succeeds, the resulting item will get all information from the item B. B will therefore govern the item A in the syntactic representation. In the DR–tree representing the process of deriving a syntactic structure this in fact means that there will be an oblique edge between the node (of the DR–tree) representing the item A and the node (of the DR–tree) representing the item X and a vertical edge between the node representing the item B and the node representing the item X, as shown in Fig. 9.1.

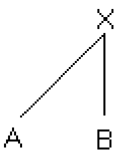

Fig. 9.1. The scheme of a simplified DR–tree for this metarule

```
}
IF A.wcl = prn THEN
{The item A represents a pronoun. If not, then the rule fails.}
        IF A.prn_type = poss THEN
{A is a possesive pronoun}
                IF B.dep_poss = yes THEN
                        B.wcl ? nonsens SecondPossModNoun
                                ELSE X.dep_poss:=yes ENDIF
```

{This part of the metarule contains the first occurrence of a soft constraint in this chapter, let us therefore remind that the soft constraints are marked by the question mark (?) in the position of the operator. This soft constraint has the following function: If any already processed possessive pronoun depends on B, the soft constraint blocks a successful application of this rule in the positive projective or positive nonprojective phase. In negative phases this soft constraint marks a syntactic inconsistency with the code SecondPossModNoun and the interpretation of this rule continues. The soft constraint using the keyword *nonsens* is in fact a small trick, which allows to overcome the problem that the syntax of metarules does not allow to use a soft constraint between IF and THEN. Would it be possible, the syntax of the instruction could look like this:

```
IF B.dep_poss ? yes SecondPossModNoun ELSE X.dep_poss:=yes ENDIF}
                        ELSE ENDIF
{If A is not a possesive pronoun, the processing continues.}
```

{The following lines handle the same problem of two adjectival modifiers of an identical type depending on the same noun for personal, reflexive, demonstrative, relative pronouns and totalizers, respectively.}

```
IF A.prn_type = pers THEN
        IF B.dep_pers = yes THEN
                B.wcl ? nonsens SecondPersModNoun
                        ELSE X.dep_pers:=yes ENDIF
                ELSE ENDIF
IF A.prn_type = refl THEN
        IF B.dep_refl = yes THEN
                B.wcl ? nonsens SecondReflModNoun
                        ELSE X.dep_refl:=yes ENDIF
                ELSE ENDIF
IF A.prn_type = dem THEN
        IF B.dep_dem = yes THEN
                B.wcl ? nonsens SecondDemModNoun
                        ELSE X.dep_dem:=yes ENDIF
                ELSE ENDIF
IF A.prn_type = rel THEN
        IF B.dep_rel = yes THEN
                B.wcl ? nonsens SecondRelModNoun
                        ELSE X.dep_rel:=yes ENDIF
                ELSE ENDIF
IF A.prn_type = tot THEN
        IF B.dep_tot = yes THEN
                B.wcl ? nonsens SecondTotModNoun
                        ELSE X.dep_tot:=yes ENDIF
                ELSE ENDIF
```

{The interrogative, indefinite and negative pronouns have the same property of uniqueness with respect to the governing noun as the above mentioned types of pronouns. Besides that they exhibit certain requirements on the mutual order of dependent words. The requirements were taken from the report [Bémová 96], where they are described in detail.

At this point it is necessary to stress that this set of conditions is in fact able to check the mutual order of pronouns depending on the same nominal governor, but solely in the projective constructions. In those phases where the general constraint on projectivity is relaxed, these conditions have a different role – they no longer check the correct order of depending pronouns, they only reduce the number of syntactic trees by imposing certain order of attaching pronouns of different types. To a certain extent they thus help avoid the duplicity of subtrees representing the nominal group even before the interpreter applies its inherent mechanism for pruning the duplicit structures (cf. the description of the software environment in Chapter 7.}

```
IF A.prn_type = que THEN
        IF B.dep_que = yes THEN
                B.wcl ? nonsens SecondQueModNoun
        ELSE X.dep_que:=yes
```

{The section handling the unacceptable order of dependent pronouns starts here. The interrogative pronoun may not precede an indefinite or negative one in the same phrase.}

```
                IF B.dep_indef = yes THEN
                        B.wcl ? nonsens  IllegCombQueIndef
                ELSE
IF B.dep_neg = yes THEN
                        B.wcl ? nonsens  IllegCombQueNeg
                        ELSE ENDIF
                ENDIF
```

{The end of the section of handling unacceptable word order of interrogative pronouns.}

```
        ENDIF
ELSE ENDIF
IF A.prn_type = indef THEN
        IF B.dep_indef = yes THEN
                B.wcl ? nonsens SecondIndefModNoun
        ELSE  X.dep_indef:=yes
```

{The section handling the unacceptable order of dependent pronouns starts here. The indefinite pronoun may not precede a totalizer or a demonstrative, negative or interrogative pronoun in the same phrase.

Let us demonstrate how these conditions work in the projective phase. Let us take the following set of items from the first sentence of our testbed:

nějaká [some/any]	každá [every]	činnost [activity]
lexf: nějaký	lexf: každý	lexf: činnost
wcl: prn	wcl: prn	wcl: noun
syntcl: adj	syntcl: adj	syntcl: noun
prn_type: indef	prn_type: tot	gender: fem
gender: fem	gender: fem	num: sg
num: sg	num: sg	case: nom
case: nom	case: nom	END
END	END	

The application of this metarule on the second and third item (*každá činnost*) will result in derivation of the following item:

```
činnost [activity]
        lexf:  činnost
        wcl:  noun
        syntcl: noun
        gender:  fem
        num:  sg
        case:  nom
        dep_tot:  yes
        END
```

This item is then subsequently combined with the first item (*nějaká*). It is an indefinite pronoun, therefore the constraints in this section are being checked. Due to the fact that the modified item representing the noun *činnost* with the already attached pronoun *každá* contains an attribute dep_tot, the condition *IF B.dep_tot = yes THEN B.wcl ? nonsens IllegCombIndefTot* either (in the positive phase) does not allow the attachment of the indefinite pronoun and causes this metarule to fail for this particular pair of items, or (in the negative phase), it will mark the syntactic inconsistency *IllegCombIndefTot* caused by the violation of this soft constraint into the resulting structure.

All subsequent sections taking care of this problem have a similar meaning.}

```
                IF B.dep_dem = yes THEN
                    B.wcl ? nonsens  IllegCombIndefDem
                ELSE
                    IF B.dep_neg = yes THEN
                        B.wcl ? nonsens  IllegCombIndefNeg
                    ELSE
                        IF B.dep_tot = yes THEN
                            B.wcl ? nonsens  IllegCombIndefTot
                        ELSE
                            IF B.dep_que = yes THEN
                                B.wcl ? nonsens  IllegCombIndefQue
                            ELSE ENDIF
                        ENDIF
                    ENDIF
                ENDIF
```

{The end of the section of handling unacceptable word order of indefinite pronouns.}

```
        ENDIF
    ELSE ENDIF
    IF A.prn_type = neg THEN
        IF B.dep_neg = yes THEN
            B.wcl ? nonsens SecondNegModNoun
        ELSE X.dep_neg:=yes
```

{The section handling the unacceptable order of dependent pronouns starts here. The negative pronoun may not precede a totalizer, indefinite or interrogative pronoun in the same phrase.}

```
                IF B.dep_indef = yes THEN
                    B.wcl ? nonsens  IllegCombNegIndef
                ELSE
                    IF B.dep_tot = yes THEN
                        B.wcl ? nonsens  IllegCombNegTot
                    ELSE
                        IF B.dep_que = yes THEN
                            B.wcl ? nonsens  IllegCombNegQue
                        ELSE ENDIF
                    ENDIF
                ENDIF
```

{The end of the section processing unacceptable word order of negative pronouns.}

```
        ENDIF
ELSE ENDIF
```

{The following ELSE branch belongs to the IF A.wcl = prn condition. If the item A is not a pronoun, the rule fails.}

```
ELSE FAIL ENDIF
```

{The second branch of the main IF statement.}

```
A.gender ? B.gender  errGender
A.num ? B.num  errNum
A.case ? B.case  errCase
```

{These soft constraints check the agreement in gender, number and case of the governing item B and the dependent item A. It might be an issue of discussion whether these three constraints should be here or whether they should immediately precede the assignment operator X:=B. In general the placement of individual constraints should follow a simple rule – in case the metarule is not applicable to a particular pair of items A and B, the processing of such a metarule should fail as soon as possible. Thus those constraints which can block the greatest number of unacceptable items A or B should be placed at the top of the metarule. In this case it is very difficult to decide on the proper place of these three constraints.}

```
OK
```

{If the interpretation of the metarule reaches this point, the rule may be applied and the new item X is created. At this point it is necessary to stress that even though the assignment X:=B was processed quite early in this metarule, it was mainly for the purpose of reference in subsequent statements. The actual item X is not created until the keyword OK is processed. Let us also repeat that the item X in fact represents a node of a DR–tree (cf. Fig 9.1). }

```
END_P
```

Metarule (2)

Prepositional group

```
PROJECTIVE
```

{This rule may be applied only in a projective way. The inner structure of Czech prepositional groups is generally projective, even though we may find several very complicated examples of prepositional groups, e.g. *s o hlavu větším přítelem* – (Lit.: with by head taller friend) – [with a friend by a head taller].}

```
A.syntcl = prep
B.syntcl = noun
```

{The item on the left–hand side is a preposition, the item on the right–hand side has syntactic properties of a noun}

```
A.case ? B.case  Case_Dis_Prep_Noun
```

{This soft constraint checks the agreement of the case required by the preposition and that of the nominal head of the group.}

```
IF B.wcl = prn THEN
        B.pprep ? yes NonprepFormOfPronoun
ELSE ENDIF
```

{In case that the item B is a pronoun with syntactic properties of a noun it is necessary to check whether it is in the correct (i.e. strong) form which allows the pronoun to be a head of a prepositional group.}

```
X:=B
```

{The item X gets all information from the item B except for the following two attributes}

```
X.syntcl := prephr
```

{Item X is now marked as a prepositional group. The change of a syntactic class of the governing item also blocks the possibility to attach a second preposition to the same noun.}

```
X.prep := A.lexf
```

{A "new attribute" (in fact it is not new, it is all the time virtually present in the data structure of the noun) will preserve the lexical value of the preposition for a later use (if any). This rule is one of many where it was necessary to decide which of the two items will be the governing one. This decision must be made at any moment when, unlike in the previous rule, the resulting item inherits data from both items A and B. Let us repeat the simple criterion applied in our system – whichever of the two items contains more information which might be required later, it is chosen as the governing one.}

```
OK
END_P
```

Metarule (3)

The attachment of a nominal modifier in the genitive case to the noun from the right hand side.

```
PROJECTIVE
```

{This metarule may be applied only in a projective way. The modifier may not be part of a nonprojective construction.}

```
IF A.syntcl = noun THEN
```

{The item A has syntactic properties of a noun.}

```
    IF B. syntcl = noun THEN
        IF B.case = gen THEN
```

{The item B is in the genitive case and it has syntactic properties of a noun.}

{The following lines handle the collision of this metarule with the following one. In case both the title (e.g. *pan Novák* – Mr. Novák) and the noun following the title are in the genitive case and they also agree in gender and number, these items could be processed by both metarules. In such a case the section below blocks the application of this metarule in favor of the following one.}

```
IF A.title = yes THEN
    IF A.case = gen THEN
        IF A.gender = B.gender THEN
            IF A.num = B.num THEN FAIL
            ELSE ENDIF
         ELSE ENDIF
     ELSE ENDIF
ELSE ENDIF
```

{End of the inserted block.}

```
        ELSE  FAIL ENDIF
    ELSE  FAIL ENDIF
ELSE FAIL ENDIF
```

{If the item A is not a noun and the item B either is not in the genitive case or it is not a noun, the application of the rule fails.}

```
IF A.rightgen = no THEN FAIL ELSE ENDIF
```

{The result of the condition A.rightgen = no in this form provides the result FALSE not only in case the value of the attribute *rightgen* is *yes*, but also in case the item A does not have any such attribute. This property of the interpreter makes it possible to introduce the concept of "virtual" attributes. It is possible to check the value of an attribute even if it is not part of a data structure in the given moment. This concept has already been used in the first metarule and is very often used for the same purpose in several other attributes.

This constraint is a hard constraint, although there is an alternative way of expressing this constraint as a soft one:

```
IF A.rightgen = no THEN A.wcl ? nonsens SecondGenModNoun ELSE ENDIF
```

The decision whether we should use the first or the second form depends on the error expectation. If we expect that the existence of the second noun in the genitive modifying the same head is likely, we would use the second form. On the other hand, if this type of error is unlikely, we had better use the first form since every soft constraint causes an increase of the number of derived structures and thus negatively influences the phases of negative parsing (it allows a wider application of the metarule and thus creates a larger number of new items).}

```
X:=A
X.rightgen := no
```

{The result of the application of this metarule is an item which corresponds to the item A (the governing word is the first noun). This item cannot be further modified by another noun in the genitive from the right–hand side.}

```
IF B.eos = yes THEN X.eos := yes ELSE ENDIF
IF B.end = yes THEN X.end := yes ELSE ENDIF
IF B.eoc = yes THEN X.eoc := yes ELSE ENDIF
```

{Let us now point out that the following explanation holds only for the projective processing of the metarule. The situation is completely different when the global projectivity constraint is relaxed. In such a case this mechanism in fact imposes certain order of attachment and thus reduces the number of derived items (cf. the comment to the first metarule).

The attributes *eos* and *eoc* indicate that the right end of the sentence or of the clause, respectively, has already been processed (this is important to know, for example, in a nested clause). The attribute *end* indicates that either the right end of a sentence or the right end of a clause has already been processed. These attributes are very important for several reasons. They preserve the information about commas and periods which had already been processed and thus disappeared from the sentence. This is crucial for example if there is a multiple nesting of clauses and a single comma serves as the right–end marker for several clauses. The information that everything between a head of the clause and its right end has been processed also helps to reduce the number of items derived in the process of parsing – some of the left–hand side modifiers (for example relative pronouns) are processed only after the right–hand side modifiers of the head have already been attached. This block of instructions is very common in those metarules, where the symbol A is the governing one. The values must be copied into the item X, otherwise they would be lost, because the item B is not the governing one.}

```
OK
END_P
```

Metarule (4)

Constructions of the type "pan Novák" (Mr. Novák)

```
PROJECTIVE
```

{This metarule may be applied only in a projective way.}

```
A.syntcl = noun
B. syntcl = noun
```

{Both items have syntactic properties of a noun.}

```
IF A.title = yes THEN
    IF A.case = voc THEN
        IF B.case = nom THEN
        ELSE
            IF B.case = voc THEN
```

{According to the official Czech grammar not only the address *pane*[voc.] *Nováku*[voc.], but also the form *pane*[voc.] *Novák*[nom.] is acceptable.}

```
            ELSE
                A.case ? nonsens          Improper_Case_Comb_Titl_Name
```

{Any other case than the nominative or the vocative (in combination of the vocative case of the item A) is reported as a syntactic inconsistency.}

```
            ENDIF
```

```
        ENDIF
    ELSE
        A.case ? B.case          Case_Disagr_Titl_Name
```

{The agreement in case is checked for all cases of the item A except for vocative, which is handled by the set of conditions above.}

```
    ENDIF
ELSE FAIL ENDIF
A.gender ? B.gender     Gender_Disagr_Titl_Name
A.num ? B.num           Num_Disagr_Titl_Name
```

{If the item A may be used as a title, it must agree not only in case, but also in gender and number with the head of the group.}

```
X:=B
```

{The governing item is the item B.}

```
X.title := no
```

{This assignment operator blocks the combinatorial explosion of parses in case of constructions where more than one title is involved – *předseda pan dr. Novák* (chairman mister Dr.Novák).}

```
OK
END_P
```

{Note that this metarule does not deal in any specific way with constructions of the type *Novák autor je lepší než Novák překladatel.* [Lit.: Novák author is better than Novák translator.] – (Novák is better as author than as translator.) This type of constructions could be dealt with purely on the lexical level simply by adding the attribute *title* to every surname in the dictionary of the system.}

9.2 Verbal modifiers

The second group of metarules is the most important one. It allows to process filling valency slots with nonprepositional inner participants and to attach free modifiers. The fifth metarule is the most complicated metarule of the entire grammar.

Metarule (5)

Filling of a slot in the valency frame from the left–hand side

{This metarule has its counterpart in the metarule for filling a slot in the valency frame from the right hand side. Since both rules are almost identical with the only difference that the item A from this metarule is the item B in the other one and vice versa, we are going to describe only this rule as a representative of both.}

{The first part of instructions handles the case when the governing word – item B – is an auxiliary verb.}

```
IF B.syntcl = aux THEN
```

{The item B is an auxiliary verb.}

```
    IF A. syntcl = v THEN
        IF A.v_form = inf THEN ELSE
            IF B.v_form = inf THEN FAIL ELSE ENDIF
```

{This condition blocks the attachment of one infinite verb as an inner participant of another infinite verb. Such a construction (a chain of infinitive verbs) would be handled by a special metarule. Let us at this point remind what does the keyword FAIL mean – when it is encountered in a particular branch of an IF statement, the computation of the metarule immediately stops without success. In other words, the keyword FAIL forces the processing of the whole metarule to stop without sucess, not only of a particular statement where it is located.}

```
            ELSE
    IF A.v_form = passinf THEN ELSE FAIL ENDIF
    ENDIF
```

{If the dependent word is a verb, it may be only an infinitive or a head of a chain of infinitives.}

```
    ELSE
    IF A.syntcl = noun THEN ELSE FAIL ENDIF
ENDIF
```

{The dependent word may also be a noun but nothing else – at least in this metarule. The other three types of candidates for frame slot fillers, namely adverbs, subordinated clauses and prepositional groups, are handled elsewhere. The adverbs are handled by the metarules described in section 9.8, subordinate clauses by metarules from the section 9.6. Let us remind that we have excluded prepositional groups from valency frames for reasons mentioned at the end of the previous chapter.}

```
ELSE
```

{This is the end of the section handling auxiliary verbs. Next section is almost identical, it handles main (autosemantic) verbs.}

```
IF B.syntcl = v THEN
```

{The item B is a main (autosemantic) verb.}

```
        IF A.syntcl = v THEN
            IF A.v_form = inf THEN ELSE
                IF A.v_form = passinf THEN ELSE FAIL ENDIF
            ENDIF
```

{If the dependent word is a verb, it may be only an infinitive or a head of a chain of infinitives combined with the passive voice.}

```
        ELSE
        IF A.syntcl = noun THEN  ELSE  FAIL ENDIF
    ENDIF
```

{In this part of the metarule only a noun may be a dependent word, but nothing else – the same comment as above applies here.}

```
ELSE
```

{This is the end of the section handling main (autosemantic) verbs. Next section handles nouns.}

```
    IF B.syntcl = noun THEN
        IF A.syntcl = noun THEN
            IF A.syntcl = v THEN ELSE
                IF A.v_form = inf THEN ELSE FAIL ENDIF
            ENDIF
        ENDIF
ELSE FAIL ENDIF
```

{If the item B has syntactic properties of a noun then the item A (dependent word) must also have syntactic properties of a noun or it can be a verb in infinitive – cf. *touha psát* [the desire to write]. Nothing else may fill the frame of a noun in a nonprepositional case in this metarule – the same comment as above applies here.}

```
    ELSE
```

{The following section handles adverbs and adjectives as governing words. }

```
        IF B.syntcl = adv THEN
        ELSE
            IF B. syntcl = adj THEN ELSE FAIL ENDIF
        ENDIF
    ENDIF
ENDIF
ENDIF
IF A.prn_type = rel THEN FAIL ELSE ENDIF
```

{The relative pronouns are handled by metarules in section 9.6.}

{The following section concerns the choice of a relevant slot in the valency frame of the governing word – the item B.}

```
P in B.frameset
```

{A temporary item P is used for the identification of the chosen slot.}

```
P.prep = 0
```

{The slot represents a nonprepositional case.}

```
IF A. syntcl = v THEN P.case ? inf verbal_actant_not_inf
```

{If the item A is a verb, it may only be in an infinitive form.}

```
ELSE A.case ? P.case case_disagr_in_the_frame
```

{If the item A is not a verb, it must have the same value of the attribute *case* as the slot in the valency frame which is represented by the temporary item P.}

```
ENDIF
```

{At this point it is necessary to check the subject–verb agreement. The constraint on case (nominative) is not sufficient for the subject, the subject must agree with the verb in person, gender and number.}

```
IF B. syntcl = v THEN
```

{The governing word – item B – is the main verb.}

```
    IF P.actant = act THEN
```

{The slot in the valency frame is a slot for an *actor*, an inner participant mostly corresponding to a subject. This constraint is based on the value of the attribute *actant* (inner participant) and not on the value of the attribute *case* (nominative) as a consequence of the fact that there are verbs in Czech which do not have a subject in nominative case, cf. *Kellyových*[gen.] *zpívali* (The Kelly family sang). In this case some types of nouns in genitive cases, citations, interjections etc. can have the role of the “syntactic nominative”.}

```
        IF B.v_form = inf THEN FAIL  ELSE
            IF B.v_form = passinf THEN FAIL ELSE
```

{The infinitive form of the verb must not have any subject. If the form of a governing verb is infinitive, the metarule fails.}

```
                IF A.gender = B.gender THEN
                    IF A.num = B.num THEN
                    ELSE
                        A.WCL ? nonsens subj_verb_disagr_num
                    ENDIF
                ELSE
                    A.WCL ? nonsens subj_verb_disagr_gend
                ENDIF
```

{The subject must agree in gender and number with the verb}

```
            ENDIF
        ENDIF
    ELSE
    ENDIF
ELSE
ENDIF
```

{In case the passive voice contains an infinitive or a chain of infinitives (for example the sentence *Zasedání mohou začít být svolávána zítra.* (The meetings can begin to be summoned tomorrow.)), it is necessary to check the agreement between the subject and the only verb that is not in the infinitive form.}

```
IF B.syntcl = v THEN
    IF B.v_form = inf THEN ELSE
        IF A.v_form = passinf THEN
            A.gender ? B.gender     Verb_pass_dissagr_gender
            A.num ? B.num           Verb_pass_dissagr_number
```

{If the item B is the main (autosemantic) verb and the item A is an infinitive taking part in a passive construction, both verbs must agree in gender and number. The item A does not, of course, have its own gender and number, this information is carried over from the verb in the passive voice through the whole chain of infinitives which are parts of the passive construction. This transfer of information is handled by instructions in this metarule below.}

```
        ELSE ENDIF
    ENDIF
ELSE ENDIF
X:=B
```

{If the metarule succeeds, the result gets all data from the item B.}

```
\ P from X.frameset
```

{This is the end of the section dealing with filling the slot in the valency frame of the item B.}

{The last block handles the transfer of the information about gender and number of a verb in passive voice through the chain of infinitives which take part in a passive construction to the main (autosemantic) verb – cf. the comments above.}

```
IF B.v_form = inf THEN
    IF A.v_form = passinf THEN
    X.gender := A.gender
    X.num := A.num
```

{If the governing word is an infinitive and the dependent one is an infinitive taking part in a passive construction then it is necessary to copy the information about the number and gender from the dependent verb to the governing one.}

```
    X.v_form := passinf
```

{It is also necessary to mark the fact that now also the governing verb takes part in a passive construction.}

```
    ELSE ENDIF
ELSE ENDIF
OK
END_P
```

Metarule (6)

Attachment of a free modifier (adjunct) from the right–hand side

```
CLOSEST
```

{This metarule should be applied only to adjacent input items A and B, in the sense of static order of input words. This means that if the input items A' A" B' B" are in this order on the input and the item A is the governing one, the metarule may be applied only to the pairs of items A"B' and A"B".

In this metarule the keyword CLOSEST blocks the possibility of an attachment of a free modifier from the subordinate clause to the main verb of the main clause or the spurious ambiguity of an attachment of a free modifier to all preceding nouns etc.}

```
IF B.syntcl = prephr THEN ELSE
    IF B.syntcl = noun THEN
        IF B.case = ins THEN ELSE ENDIF
```

{The free modifier is either a prepositional group or a noun in the instrumental case. The instrumental case is not the only nonprepositional case which may have the role of a free modifier. No other cases are taken into account because it is impossible to distinguish when they are in a role of an inner participant or a free modifier on the basis of purely syntactic clues (cf. the discussion at the end of the previous chapter). Other nonprepositional cases are quite rare in the syntactic role of a free modifier compared to instrumental case; thus it is possible to neglect them. Would all the other cases be allowed as free modifiers we would get spurious ambiguities everywhere – all nonprepositional cases would be taken both as filling the slot of an inner participant and of a free modifier. The interpreter of our metarules does not allow to formulate a constraint "If there is a suitable valency slot then the item under consideration is an inner participant, otherwise it is a free modifier."}

```
        ELSE
            IF B.syntcl = select THEN ELSE FAIL ENDIF
    {A free modifier may also be a selective construction (čtyři z pěti zubních lékařů – four out of
five dentists.)}
        ENDIF
    ENDIF

    IF A.syntcl = v THEN ELSE
    {The free modifier may depend on a verb or on a noun.}
        IF A.syntcl = noun THEN
    {If the free modifier depends on a noun, it is necessary to block the spurious ambiguity of an
application of this metarule and of the metarule handling constructions of the type "chairman Mao"
(the metarule (4)). This situation is similar to that from the second metarule and the solution proposed
is identical – an introduction and testing of a special auxiliary attribute.}
            IF A.titul = yes THEN
                IF A.case = B.case THEN
                    IF A.gender = B.gender THEN
                        IF A.num = B.num THEN FAIL ELSE ENDIF
                    ELSE ENDIF
                ELSE ENDIF
            ELSE
                IF B.wcl = num THEN FAIL ELSE ENDIF
    {In the role of a free modifier a numeral may depend only on a verb.}
            ENDIF
        ELSE FAIL ENDIF
    ENDIF

    IF B.nofm = yes THEN FAIL ELSE ENDIF
    {This condition blocks the attachment of free modifiers in case for some reason the item B
cannot be attached as a free modifier. }
    X := A
    {The item A is a governing word.}
    IF A.syntcl = noun THEN X.rightgen := no ELSE ENDIF
    {An attachment of a free modifier to the noun from the right–hand side blocks the possibility
of an attachment of a nominal modifier in the genitive case to the same noun.}
    IF B.eos = yes THEN X.eos := yes ELSE ENDIF
    IF B.end = yes THEN X.end := yes ELSE ENDIF
    IF B.eoc = yes THEN X.eoc := yes ELSE ENDIF
    {The information that the end of a clause or the end of the sentence has already been
processed is carried over from the dependent to the governing item – cf. the metarule (3).}
    OK
    END_P
```

9.3 Complex verbal forms

The third group of metarules concerns the complex verbal forms, namely the conditional and past tense forms. The rules are much simpler than the rules of the previous group. These metarules generally combine auxiliaries with main verbs and block the attachment of incorrect number of auxiliaries to the main verb. They do not (and cannot) take care for a proper position of clitics in clauses.

Metarule (7)

Conditional auxiliary to the left of the main verb

{This metarule handles the compound present and past conditional form of the verb composed of the auxiliary in the conditional form followed by the main (autosemantic) verb with a possible additional auxiliary verb in the past tense in between (for the past conditional). This metarule has its counterpart (not presented here) in the metarule handling the same phenomenon with the reversed order of words. Both metarules are almost identical, the difference is only in the order of words.}

```
A.syntcl = aux
B.syntcl = v
B.v_form ? pastp WrongVerbalFormofCondMood
{The item B has syntactic properties of a verb in the past participle.}
X := B
{The governing item is the item B.}
IF A.mode = cond THEN
{The item A is an auxiliary verb in the conditional mood.}
        IF B.cond_aux = no THEN
                B.wcl ? nonsens   Two_cond_aux
{The value of an attribute cond_aux indicates whether the auxiliary verb in the conditional mood has already been attached to the main verb.}
        ELSE ENDIF
        X.v_cl := cond
{The attribute v_cl of the item X receives the value indicating that the item represents a compound verb in conditional}
        X.cond_aux := no
{This statement blocks the attachment of another auxiliary in the conditional mood to the same main verb.}
ELSE   IF A.v_form = pastp THEN
{The auxiliary verb is in the past form – it is a part of the past conditional.}
                X.past_aux := yes
        ELSE FAIL ENDIF
{If the auxiliary verb is not in the conditional or past form, the metarule is not applied.}
OK
END_P
```

Metarule (8)

Nominal predicate with the copula

```
A.syntcl = aux
{The item A represents an auxiliary verb.}
IF B.syntcl = adj THEN
ELSE IF B.syntcl = noun THEN ELSE FAIL ENDIF
ENDIF
{The item B has either syntactic properties of an adjective or of a noun.}
IF B.case = nom THEN
ELSE    B.case ? ins  Wrong_case_by_copula
ENDIF
{The nominal predicate may be only in the nominative or instrumental case.}
X := A
{The governing item is the copula.}
X.syntcl := v
```

{This assignment of the new value to the attribute *syntcl* has two functions. The first function is to avoid the attachment of another noun or adjective as a nominal predicate to the same copula. The second function is to indicate that the predicate is complete and the processing of the rest of the sentence may continue (as in the previous cases, this comment concerns the projective phases).}

OK
END_P

Metarule (9)

Complex past tense

{This metarule handles the combination of the auxiliary verb and the main (autosemantic) verb in the past tense. The grammar contains also the metarule handling the same combination of words in the reversed order.}

```
A.syntcl = aux
A.tense = pres
```
{The item A represents an auxiliary verb in the present tense (*dělal* ***jsem*** – I *was* doing).}

```
B.v_form = pastp
```
{The item B represents a verb in the past tense.}

```
IF B.depaux = yes THEN
        B.depaux ? nonsens  Second_aux
```
{The item B cannot have one auxiliary already attached.}

```
ELSE ENDIF
X := B
```
{The governing item is the main (autosemantic) verb.}

```
X.depaux := yes
```
{This technical attribute blocks the attachment of the second auxiliary.}

OK
END_P

Metarule (10)

Verb in a passive voice combined with a copula from the left–hand side

{This metarule has its counterpart which differs only in the mutual position of items – the copula is on the right–hand side.}

```
A.lexf = být
```
{The lexical value of the lemma of the copula is checked.}

```
A.syntcl = aux
```
{The item A is an auxiliary verb. It may seem that this test is not necessary due to the previous condition, but it is necessary to take into account that the auxiliary verb *být* [to be] may play several roles. It is therefore represented by several unambiguous items, out of which only those with a syntactic role of an auxiliary verb are to be used here.}

```
IF A.mode = cond THEN FAIL ELSE ENDIF
```
{Conditional mood will be handled by a special metarule.}

```
B.syntcl = v
B.pass = per
```
{The item B is a verb in the passive form.}

```
X := B
```
{The governing item is the passive verb.}

```
X.v_form := passinf
```
{The newly created item is marked as being in the passive voice.}

OK
END_P

Metarule (11)

Unattached single verb in the infinitive form being dangling in the clause

NEGATIVE

{This metarule is a first example of an error anticipation metarule. It is applied only in negative phases, otherwise it is ignored by the interpreter. The item derived as a result of the application of this metarule is marked as containing a syntactic inconsistency.}

```
A.syntcl = v
B.syntcl = v
A.v_form = fin
B.v_form = inf
```

{Both input items are verbs, one in the finite and the other in the infinite form}

```
X := A
```

{The verb in the finite form is the governing item.}

```
IF B.eos = yes THEN X.eos := yes  ELSE ENDIF
IF B.end = yes THEN X.end := yes  ELSE ENDIF
IF B.eoc = yes THEN X.eoc := yes  ELSE ENDIF
```

{The information that the end of the sentence or clause has already been processed is carried over from the dependent to the governing item – cf. the comments on the metarule (3).}

```
OK
END_P
```

9.4 Coordination

The fourth group of metarules handles some examples of coordination, namely the coordination of nouns and verbs in the infinite form and the coordination of clauses. It is interesting for at least two reasons. First, it contains a section handling the coordination of nouns in the nominative case, which is very important for a correct analysis of sentences with coordinated subjects where individual members of the coordination are of a different gender. In Czech this phenomenon requires very complicated rules. Second, this group of rules contains a second example of an error anticipating metarule, which is applied only in case the positive projective phase fails to build a complete tree.

Metarule (12)

Coordination of nouns, first part

{This metarule represents the first example of how complex problems of interaction of more than two items may be handled in the metagrammar. The interaction is achieved through the use of a special (unique) value of the attribute *syntcl* assigned to the item representing the result of the first rule of the sequence (cf. the comments inside the metarule).

This metarule is the first part of a sequence of two metarules handling the coordination of nouns. This type of coordination is handled separately from other types of coordination because it has to follow a complex set of agreement rules when it becomes a subject.}

```
A.syntcl = conj
A.conjtype = coord
```

{The item A is a coordinate conjuction.}

```
B.syntcl = noun
```

{The item B has syntactic properties of a noun.}

```
X := B
```

{The item B contains more information necessary for the future processing, therefore it is taken as the governing word.}

```
X.syntcl:= koordn
```

{The value of the attribute syntcl is changed. The new value is unique – it is used only in this metarule and in the following one – and thus serves as a connection between the metarules handling the same problem. This technique is used in all cases where more than one metarule is necessary for handling a certain syntactic phenomenon.}

```
OK
END_P
```

Metarule (13)

Coordination of nouns, second part

{This metarule is the second metarule of the pair handling nominal coordination. It is an open question if the metarule should not be applied with the keyword CLOSEST in a similar manner as the attachment of free modifiers. If a more complicated nominal groups are coordinated, it is in many cases impossible to distinguish syntactically which items should be coordinated. This results in spurious ambiguities as for example in the third sample sentence of our testbed. The question whether it is acceptable to apply this metarule only for the nearest possible candidates, would probably require a deeper linguistic research. For the time being we prefer deriving all variants.}

```
A.syntcl = noun
```

{The item A has syntactic properties of a noun.}

```
B.syntcl = koordn
```

{The item B was created as a result of an application of the previous metarule.}

```
A.case ? B.case  Wrong_case_in_coord
```

{The coordination of nouns is applicable only when both are in the same case.}

```
X := A
```

{The item A is copied into X. This assignment instruction has to precede the following section because the item X must be defined before its attributes may be assigned a value.}

```
X.num := pl
```

{The value of an attribute num of the item representing the whole coordination is set to plural. Unfortunately, in Czech it is possible in this case to assign also a singular number to the coordination of this type. That is very inconvenient for our system, because the interpreter does not make it possible to create two variants of output items for a single metarule. For this reason it was necessary to add another metarule into the metagrammar, differing only in this place – the assignment is replaced by the following nested condition:

```
IF A.num = sg THEN
    IF B.num = sg THEN ELSE FAIL ENDIF
ELSE FAIL ENDIF
```

which blocks the application of this rule in case either of items A and B is in plural and thus it is fully covered by this rule and no variant is necessary.}

```
IF A.case = nom THEN
```

{This is a slight simplification exploiting the fact that the vast majority of subjects is in the nominative case. Due to the fact that this metarule handles only the coordination of nouns, no other possible types of subjects (infinitives, subordinated clauses) are taken into account here.}

```
    IF A.gender = anim THEN ELSE
```

{If the gender of the item on the left–hand side is masculine animate, it is not necessary to do anything since the value is copied by default by the assignment X := A.}

```
        IF B.gender = anim THEN
            X.gender:= anim
```

{If either of the items A and B is masculine animate, then the result should also be masculine animate.}

```
        ELSE
        IF A.gender = neut THEN
            IF B.gender = neut THEN
```

```
                IF A.num = sg THEN
                    X.gender := fem
                ELSE    IF B.num = sg THEN
                            X.gender := fem
                        ELSE    X.gender := B.gender
                        ENDIF
                ENDIF
ELSE X.gender:= B.gender
```

{If both items A and B are neuter and at least one of them is in singular, then the whole coordination receives the feminine gender, otherwise it receives the same gender as the item B. (In this branch of the IF statement neither item may be masculine animate.) If only A is in neuter, then the whole coordination retains the gender of the item B.}

```
            ENDIF
        ELSE ENDIF
        ENDIF
    ENDIF
ELSE ENDIF
IF B.eos = yes THEN X.eos := yes ELSE ENDIF
IF B.end = yes THEN X.end := yes ELSE ENDIF
IF B.eoc = yes THEN X.eoc := yes ELSE ENDIF
```

{Standard way of the transfer of the information that the end of a clause has already been processed from the dependent to the governing item – cf. the comments on the third metarule.}

```
OK
END_P
```

Metarule (14)

Coordination with ill–placed comma

{This is another example of an error anticipation metarule. }

```
NEGATIVE
```

{This keyword marks an error anticipation metarule.}

```
A.syntcl = comma
```

{The item A is a comma.}

```
IF B.syntcl = koordinf THEN ELSE
    IF B.syntcl = koordn THEN ELSE FAIL ENDIF
ENDIF
```

{The item B is a coordination of nouns or verbs in infinite form. This is the place where also other types of coordination may be handled in future by means of adding other nested conditions.}

```
X := B
OK
END_P
```

Metarule (15)

Coordination of infinitive verbs, the first part

```
A.wcl = conj
A.conjtype = coord
```

{The item A is a coordinate conjuction.}

```
B.syntcl = v
B.v_form = inf
```

{The item B is a verb in the infinitive form.}

```
X := B
```

```
{B is the governing item.}
X.syntcl:= koordinf
{The result is assigned a unique value for its syntactic class, allowing to use the item solely in the next metarule.}
OK
END_P
```

Metarule (16)

Coordination of infinitive verbs, the second part

```
A.syntcl = v
A.v_form = inf
{The item A is a verb in the infinitive form.}
B.syntcl = koordinf
{The item B was derived as a result of an application of the previous metarule.}
X := A
{A is the governing item.}
IF B.eos = yes THEN X.eos := yes ELSE ENDIF
IF B.eoc = yes THEN X.eoc := yes ELSE ENDIF
IF B.end = yes THEN X.end := yes ELSE ENDIF
{Standard way of the transfer of the information that the end of a clause has already been processed from the dependent to the governing item – cf. the comments to the metarule (3).}
OK
END_P
```

Metarule (17)

Coordination of clauses, first part

```
{This is the first metarule from the pair of metarules handling the coordination of clauses. It combines the conjunction with the clause on its right–hand side.}
A.syntcl = conj
A.conjtype = coord
{The item A represents a coordinate conjunction.}
IF B.syntcl = v THEN ELSE
        IF B.syntcl = aux THEN ELSE FAIL ENDIF
ENDIF
{The item B is either a main (autosemantic) or an auxiliary verb.}
B.eos = yes
{It is necessary to combine both items only in case the rest of the clause has already been processed.}
X := B
{The item B is the governing item. This is again an arbitrary decision because the only information required in the further processing is contained in the new value of the attribute syntcl.}
X.syntcl := clcoord
{The value of the attribute syntcl of the newly created item is set to a unique value marking the fact that the item represents the combination of the coordinate conjunction and the second clause of the coordination – cf. the comments to the metarule (12).}
OK
END_P
```

Metarule (18)

Coordination of clauses, second part

{This is the second metarule from the pair of metarules handling the coordination of clauses. It combines the first clause with the rest of the coordination.}

```
A.syntcl = v
```

{The item A is a verb.}

```
IF A.v_form = inf THEN FAIL ELSE ENDIF
IF B.v_form = inf THEN FAIL ELSE ENDIF
```

{This metarule should not be applied for the coordination of infinite verbs, this phenomenon is handled by the metarules (15) and (16).}

```
IF B.wcl = vb THEN
        IF B.syntcl = clcoord THEN ELSE
                IF B.syntcl = comv THEN
```

{It is necessary to handle the coordination with a comma (cf. the following metarule, in which the item with *syntcl=comv* is created) in the same manner as the coordination with a coordinate conjunction due to the possibility of more than two clauses being coordinated.}

```
                        IF B.rel_clause = yes THEN FAIL ELSE ENDIF
```

{The relative clauses cannot be handled by this metarule, relative clauses are treated separately due to the necessity to check the agreement of the relative pronoun and the noun it is referring to.}

```
                ELSE FAIL ENDIF
        ENDIF
ELSE FAIL ENDIF
X := A
```

{The item A is the governing item.}

```
IF B.eoc = yes THEN X.eoc := yes  ELSE ENDIF
IF B.eos = yes THEN X.eos := yes  ELSE ENDIF
IF B.end = yes THEN X.end := yes  ELSE ENDIF
```

{The information that the end of the clause has already been processed is transferred – cf. the comments to the third metarule.}

```
OK
END_P
```

9.5 Embedded nominal groups and clauses

This section contains four metarules handling the problem of embedded nominal groups and clauses. At this point it is necessary to stress that we are using the term „embedded" in a slightly different manner than the traditional terminology. We distinguish between a general term of a subordinated clause (phrase, group) and a more specific term of an embedded clause (phrase, group). The latter term, as we understand it, represents subordinated clauses (phrases, groups) which are inserted into the main clause in such a manner that there is a non–empty tail of the main clause following the embedded clause (phrase, group). In other words, the embedded section of a sentence divides the main clause into two non–empty parts.

Metarule (19)

Embedded nominal group depending on a noun, first part

{The construction described by the metarules in this section consists of four words – a governing noun, comma, a nominal (or verbal) head of the embedded group (clause) and the comma following the embedded group (clause). This metarule handles the left comma and the head of the embedded nominal group. The main problem is that it is not possible to restrict the use of this particular metarule only to the construction indicated above. The pair comma – noun may also be a part of other constructions, for example a coordination of more than two nominal elements, where usually the first *n–1* elements are separated by commas and only the last pair of nouns has a conjunction in between.}

```
A.syntcl = comma
IF B.syntcl = noun THEN ELSE
    IF B.syntcl = v THEN ELSE FAIL ENDIF
ENDIF
```
{The item A is a comma and the item B has syntactic properties of a noun or of a verb.}

```
IF B.prn_type = rel THEN FAIL ELSE ENDIF
```
{Relative clauses are handled by the metarules in section 9.6. }

```
X := B
IF B.syntcl = noun THEN X.syntcl := comn ELSE ENDIF
IF B.syntcl = v THEN X.syntcl := comv ELSE ENDIF
```
{The item B is the governing one. The item X is assigned a special syntactic class, which keeps track of the fact that the comma to the left has already been processed.}

```
OK
END_P
```

Metarule (20)

Embedded noun or clause depending on a noun, second part

{Similarly as the first metarule of this set, this metarule is applicable not only to embedded nominal groups, but also to embedded clauses, because there is no particular reason to handle the two types separately.}

```
IF A.syntcl = comn THEN ELSE
    IF A.syntcl = comv THEN ELSE FAIL ENDIF
ENDIF
```
{The item A represents either a comma and a noun, or a comma and a clause.}

```
B.syntcl = comma
```
{The item B is a comma following a noun phrase or a clause. This comma must be processed before the noun phrase or a clause is attached to the preceding noun, otherwise it might happen that the words belonging to the embedded phrase or clause will interact with the words of the governing clause (e.g. fill the slots in the frame etc.). This remark of course applies only to the projective interpretation of the metagrammar, in the nonprojective one this simple restriction does not work, cf. the comments on the first and third metarules.}

```
X := A
X.syntcl := comncom
```
{Since the item A contains more information than the item B (comma), it is chosen as the governing item. Its attribute syntcl is assigned a special value indicating that it represents an embedded phrase or clause.}

```
X.eoc := yes
X.end := yes
```
{The information that the comma following the nominal group or the clause has already been processed is stored in the attribute eoc and end. The attribute eos is not used, because its role is to indicate that the final period, question mark or exclamation mark have already been processed, i.e. it indicates that the end of the whole sentence has already been reached – cf. the comments on the third metarule.}

```
OK
END_P
```

Metarule (21)

Embedded noun or clause depending on a noun, third part

{This is the last part of the section handling embedded phrases or clauses. This rule has to take into account both possibilities – that the comma following the phrase or the clause has either already been processed or not, the latter in case it belongs, for example, also to another subordinate clause in case of multiple embedding etc.}

```
A.syntcl = noun
```

{The item A has syntactic properties of a noun.}

```
IF B.syntcl = comncom THEN ELSE
```

{If the item B was created as a result of the previous metarule, then everything is OK.}

```
    IF B.syntcl = comn THEN
        B.end ? yes   Incomplete_clause
    ELSE FAIL ENDIF
ENDIF
```

{If the item B has not been created by the previous metarule, then it is OK only in case its attribute end indicates that the comma following the phrase or the clause has for some reason been already processed.}

```
X := A
```

{The antecedent is the governing item.}

```
IF B.eos = yes THEN X.eos := yes ELSE ENDIF
IF B.eoc = yes THEN X.eoc := yes ELSE ENDIF
IF B.end = yes THEN X.end := yes ELSE ENDIF
```

{The information that the comma following the embedded nominal group or clause has already been processed is transferred – cf. the comments on the third metarule.}

```
X.rightgen := no
```

{The embedded clause or phrase blocks the possibility of an attachment of the noun in the genitive case in apposition – cf. the comments on the relevant part of the third metarule.}

```
OK
END_P
```

9.6 Subordinate clauses

This section contains metarules handling subordinate clauses, including relative ones.

Metarule (21)

Filling the slot in the valency frame by a relative pronoun from the left–hand side

{This metarule does not have a counterpart handling the same problem from the right–hand side because the relative pronoun in a relative clause must precede the verb.}

```
IF B.syntcl = v THEN
    B.end ? yes  Unf_rel_cl
ELSE FAIL ENDIF
```

{The item B is a main (autosemantic) verb with a complete set of right–hand side modifiers already attached. }

```
A.syntcl = noun
A.prn_type = rel
```

{The item A is a relative pronoun. Since relative pronouns are the only words having the value of the attribute prn_type equal to rel, it is not necessary to check any other attribute.}

```
P in B.frameset
```

{A temporary item P represents one slot of the frame of the item B (verb).}

```
P.prep = 0
```

{The slot must not require a preposition.}

```
A.case ? P.case    Case_disagr_in_the_frame
```

{The pronoun must agree with the item P in case. This is a standard constraint for a filler of the slot in valency frame.}

{The following block of constraints takes care of the subject–predicate agreement similarly as in the fourth metarule.}

```
IF P.actant = act THEN
    IF B.v_form = inf THEN FAIL ELSE
        IF B.v_form = passinf THEN FAIL ELSE
```

{If the chosen slot is a slot for an actor and the item B (verb) is in the infinitive form, this metarule may not be applied – it fails because the verb in an infinitive form does not have a subject(actor).}

{If the item B is not a verb in the infinitive form then it is necessary to check the agreement in gender and number of both items.}

```
            IF A.gender = B. gender THEN
                IF A.num = B.num THEN ELSE
                    A.wcl ? nonsens subject_verb_disagr
```

{Soft constraint handling the syntactic inconsistency in number between the predicate and subject (actor).}

```
                ENDIF
            ELSE
                A.WCL ? nonsens subject_verb_disagr
```

{Soft constraint handling the syntactic inconsistency in gender between the predicate and subject (actor).}

```
            ENDIF
        ENDIF
    ENDIF
ELSE
```

{If the chosen slot is not a slot for an actor, the processing of the metarule continues.}

```
ENDIF
X := B
```

{The item B is a dominant item.}

```
\ P from X.frameset
```

{The section dealing with slots in the verbal frame ends here.}

```
X.syntcl := rel_clause
```

{The newly created item X is marked as the item representing the relative clause.}

```
X.relgender := A.gender
X.relnum := A.num
```

{The gender and number of the relative pronoun is kept for future use. In the syntactic tree representing the input sentence it means that both gender and number of the relative pronoun are propagated bottom up through the tree.}

```
OK
END_P
```

Metarule (23)

The attachment of a relative subordinate clause – first part

{This part of the metarule attaches the comma preceding the relative clause to its governing verb.}

```
A.syntcl = comma
B.syntcl = rel_clause
```

{The item A is a comma, the item B is a relative clause. The use of a special value rel_clause is necessary in order to avoid the processing of general embedded or subordinate clauses by this metarule.}

```
X := B
```

{The item B is dominant}

```
X.syntcl := com_rel_clause
```

{This special value of the attribute *syntcl* of the result contains the information that the comma preceding the relative clause has already been processed.}

```
OK
END_P
```

Metarule (24)

The attachment of a relative subordinate clause – second part

{This part of the metarule handles the attachment of the relative clause to the preceding noun.}

```
A.syntcl = noun
```

{The item A has syntactic properties of a noun}

```
IF B.syntcl = com_rel_clause THEN
ELSE IF B.syntcl = rel_clause THEN
                    B.syntcl ? nonsens Missing_comma_prec_rel_cl
```

{This branch handles the situation where there is no comma preceding the relative clause.}

```
        ELSE FAIL ENDIF
ENDIF
B.relgender ? A.gender          Wrong_relgender
B.relnum ? A.num                Wrong_relnum
```

{The gender and number of the item A and the gender and number of the relative pronoun must agree. The attributes relnum and relgender refer to the properties of a relative pronoun while B.num and B.gender refer to properties of a main (autosemantic) verb of the relative clause and they are irrelevant here.}

```
X := A
```

{The item A is dominant.}

```
X.rightgen := no
```

{The attached relative clause blocks the possibility of the modification by a noun in genitive case.}

```
IF B.eos = yes THEN X.eos := yes ELSE ENDIF
IF B.eoc = yes THEN X.eoc := yes ELSE ENDIF
IF B.end = yes THEN X.end := yes ELSE ENDIF
```

{The whole relative clause has already been processed – the value of the attributes *eos, eoc* and *end* was set when the (possible) end of the clause was processed – cf. the comments on the third metarule.}

```
OK
END_P
```

Metarule (25)

The attachment of a subordinate clause by a subordinate conjunction (the first part).

{This is the first metarule from the set of three metarules handling this type of construction. It combines the head of the subordinate clause with the subordinate conjunction.}

```
A.syntcl = conj
A.conjtype = subord
```

{The item A represents a subordinate conjunction.}

```
A.conjcl = cls
```

{The conjunction may be used for connecting the clauses.}

```
IF B.syntcl = v THEN
ELSE    IF B.syntcl = aux THEN ELSE FAIL ENDIF
ENDIF
```

{The item B represents either a verb or an auxiliary verb. It is supposed to be the head of the subordinate clause.}

X := A

{The governing item is the conjunction. This is an arbitrary choice since the only information necessary for further processing is contained in the new value of the *syntcl* attribute. No other information is being propagated through the tree.}

X.syntcl := clsub

{This value indicates that the newly created item represents the subordinate clause together with the subordinate conjunction.}

IF B.eos = yes THEN X.eos:=yes ELSE ENDIF
IF B.eoc = yes THEN X.eoc:=yes ELSE ENDIF
IF B.end = yes THEN X.end:=yes ELSE ENDIF

{Standard propagation of the information that the end of the sentence (clause) has already been processed – cf. the comments to the metarule (3).}

OK
END_P

Metarule (26)

The attachment of a subordinate clause by a subordinate conjunction (the second part).

{This is the second metarule from the set of three metarules handling this type of construction. It combines a comma and the subordinate conjunction from the previous metarule.}

A.syntcl = comma

{The item A represents a comma.}

B.syntcl = clsub

{The item B was created as a result of the application of the previous metarule.}

B.end ? yes Missing_end

{Soft constraint checking whether the end of the sentence (clause) has already been processed.}

X := A

{The comma is the governing item of this metarule. This is again an arbitrary choice. Similarly as in the previous metarule, no other information than that contained in the new value of the attribute syntcl is necessary for further processing.}

X.syntcl := clscomm

{This value indicates that the newly created item represents the subordinate clause combined with the conjunction and the comma.}

IF B.eos = yes THEN X.eos:=yes ELSE ENDIF
IF B.eoc = yes THEN X.eoc:=yes ELSE ENDIF
IF B.end = yes THEN X.end:=yes ELSE ENDIF

{Standard propagation of the information that the end of the sentence (clause) has already been processed – cf. the comments on the metarule (3).}

OK
END_P

Metarule (27)

The attachment of a subordinate clause by a subordinate conjunction (the third part).

{This is the last metarule from the set of three metarules handling this type of construction. It combines a verb (head of the main clause) with the subordinate conjunction from the previous metarule.}

A.syntcl = v

{The item A represents a verb.}

IF B.syntcl = clscomm THEN

{If the item B represents an item created by the previous metarule then everything is O.K.}

```
ELSE   IF B.syntcl = clsub THEN
                B.syntcl ? nonsens  Missing_comma_prec_subcl
```

{If the item B represents an item created by the metarule (24) then there is a missing comma between the verb in the main clause and the subordinate clause.}

```
       ELSE FAIL ENDIF
ENDIF
B.end = yes
```

{Only the items representing a complete subordinate clause may be used.}

```
X := A
```

{The verb is the governing item of the whole construction.}

```
IF B.eos = yes THEN X.eos:=yes ELSE ENDIF
IF B.eoc = yes THEN X.eoc:=yes ELSE ENDIF
IF B.end = yes THEN X.end:=yes ELSE ENDIF
```

{Standard propagation of the information that the end of the sentence (clause) has already been processed – cf. the comments on the metarule (3). In this metarule it might seem that the value of eoc (indicating the end of the clause) should not be propagated further due to fact that the complete subordinated clause has already been attached to the main clause. Unfortunately, the situation may not be as easy as that because the comma following the subordinated clause may serve as the final comma for several clauses; therefore it is necessary to preserve also the value of the attribute eoc.}

```
OK
END_P
```

9.7 Brackets

This section contains very simple metarules handling brackets and their content. The treatment is similar to the treatment of embedded clauses, with some simplifications. Generally, the problem of brackets is much simpler due to the fact that in syntactically correct sentences they appear in pairs and they clearly indicate which member of the pair is the opening bracket and which is the closing one. Unlike with commas, even if brackets are nested, the number of the opening ones equals the number of the closing ones. On the other hand, there is no restriction on what the content of the brackets may be. It may be anything from a single character to several sentences. For this reason we have decided not to parse the content of brackets and only to attach every single item to the opening bracket. We are fully aware of the fact that this solution is definitely not the ideal one. It would be much better to extract the content of brackets before parsing. The content could then be parsed in a standard manner and the result included into the syntactic structure of the parsed sentence without brackets, if necessary.

Metarule (28)

The attachment of the content and the closing bracket to the opening bracket

```
PROJECTIVE
```

{Due to the simplification of parsing the content of brackets it has no sense to apply this metarule in a nonprojective manner.}

```
A.syntcl = opening
```

{The item A is an opening bracket, the item B may be anything.}

```
X := A
```

{The opening bracket is the governing item.}

```
IF B.syntcl = closing THEN X.syntcl:=bracket ELSE ENDIF
```

{Temporary value of syntcl is used as a marker that a complete content followed by a closing bracket has already been processed.}

```
OK
END_P
```

Metarule (29)

The attachment of the opening bracket to the immediately preceding item

```
CLOSEST
{This keyword takes care of the attachment to the immediately preceding item as the only candidate.}
B.syntcl = bracket
{No constraints on A, while the item B is a result of the application of the previous metarule.}
X := A
{Quite naturally, the item A is the governing one.}
OK
END_P
```

9.8 Adverbs

There are two simple metarules in this section, representing two ways of attachment of adverbs.

Metarule (30)

An attachment of the adverb from the left

{The metagrammar contains also a counterpart of this metarule, namely the metarule for the attachment of an adverb to a governing item from the right–hand side. Due to its similarity with this metarule we do not describe it here.}

```
A.syntcl = adv
{The item A is an adverb.}
{The following set of constraints handles an attachment of the adverb to the word of the correct type according to the value of the attribute addep.}
IF B.syntcl = v THEN
        A.addep ? advb WrongTypeOfAdverbWithVerb
ELSE
        IF B.syntcl = adj  THEN
                A.addep ? ada WrongTypeOfAdverbWithAdjective
        ELSE
                IF B.syntcl = adv  THEN
                        A.addep ? add WrongTypeOfAdverbWithAdverb
                ELSE
                        IF B.syntcl = noun  THEN
                                A.addep ? adnl WrongTypeOfAdverbWithNoun
                        ELSE
                                IF B.syntcl = prephr THEN
                                        A.addep ? adnl  WrongTypeOfAdverbWithNoun
                                ELSE FAIL ENDIF
                        ENDIF
                ENDIF
        ENDIF
 ENDIF
X:=B
{The item B is the governing word.}
IF A.pair = left THEN X.pair := left ELSE ENDIF
```

{The attribute pair indicates that the adverb may belong to an expression like *nejen – ale i* [not only – but also]. Its value is propagated through the syntactic tree until it is needed for the completion of the expression – cf. the comments in section 9.12.}

OK
END_P

Metarule (31)

The attachment of an adverb to a conjunction from the right–hand side

{The raison d`être for this metarule is the construction *i nadále* [and henceforth] and other similar constructions.}

B.syntcl = adv
{The item B represents an adverb.}

A.syntcl = conj
{The item A is a conjunction.}

X := B
{The item B is the governing one.}

OK
END_P

9.9 Composition of numerals

The metarule in this section handles the composition of basic numerals in the text. There are no semantic constraints applied, so the number *three thousand five hundred twenty one* will be parsed in the same manner as a chain of numbers *one three five thousand hundred twenty*.

Metarule (32)

Composition of basic numerals (numbers)

A.wcl = num
A.numtype = dig
A.numcl = ord
{The item A represents a basic numeral.}

B.wcl = num
B.numtype = dig
B.numcl = ord
{The item B represents a basic numeral too.}

A.gender ? B.gender	Disagr_gender
A.num ? B.num	Disagr_num
A.case ? B.case	Disagr_case

{Both numerals must agree in gender, number and case.}

X := B

{The second numeral is the governing item. This is necessary because the case of the noun following the number (composed of basic numerals expressing digits) depends in the nominative and accusative case on the last digit in the number (cf. *dvacet jeden den* vs. *dvacet pět dnů* [twenty one days$_{[nom.]}$ vs. twenty five days$_{[gen.]}$]) Let us stress that this metarule deals with the composition of numerals, it does not deal with attaching the noun following the numeral to the numeral, therefore it is not necessary to include here handling the special cases of small Czech numerals (1,2,3 and 4) which behave in a different manner than the remaining numerals.}.

OK
END_P

9.10 Selective construction

The selective constructions are the first example of metarules which cover less frequent and rather specialized constructions of the language. The range of phenomena covered by subsequent sections is definitely not complete from the point of view of the coverage of all subtle properties, special constructions and exceptions present in the language. On the other hand, we have tried to demonstrate on this limited set of examples various modes of how our interpreter makes it possible to solve even rather infrequent constructions of various types; we also want to demonstrate some constructions which definitely require alternative approaches.

Metarule (33)

The selective construction containing adjectives in superlative or comparative and numerals. There is one more group of words (certain pronouns, e.g. *každý* [every], *žádný*, *někdo* [somebody], *nikdo* [nobody] etc.) which may take part in this type of construction, but this metarule does not cover them. The reason is that these words do not establish a compact group, it would either be necessary to enumerate them here or to assign them a special attribute with a value indicating that the particular word may take part in this construction.

{This metarule handles the constructions of the type *nejmladší z dětí* (the youngest among the children) or *čtvrtý z deseti* (the fourth from ten).}

```
A.syntcl = adj
{The item A has syntactic properties of an adjective}
B.syntcl = prephr
B.prep = z
{The item B represents a prepositional group with the preposition z [from].}
IF A.wcl = adj THEN
    IF A.deg = sup THEN ELSE
        IF A.deg = comp THEN ELSE FAIL ENDIF
    ENDIF
{If the item A is an adjective then its degree must be superlative or comparative.}
ELSE
    IF A.wcl = num THEN ELSE FAIL ENDIF
{If the item A is a numeral then no other constraints are required.}
ENDIF
X := A
{The item A is a governing item.}
X.syntcl := select
{The newly created item is assigned a special value of the attribute syntcl, which indicates that the item represents a selective construction.}
IF B.eos = yes THEN X.eos := yes ELSE ENDIF
IF B.end = yes THEN X.end := yes ELSE ENDIF
IF B.eoc = yes THEN X.eoc := yes ELSE ENDIF
{The information that the end of the clause (sentence) has already been processed is further transferred – cf. the comments in the metarule (3).}
OK
END_P
```

9.11 Comparison by means of the conjunction *než* (than)

This section contains only three metarules. The first one is more complicated, since it has to distinguish between two different types of comparative constructions with the conjunction *než* (than), namely the comparison with a noun or with a clause. This distinction is important for the second metarule of the pair.

Metarule (34)

Comparison by means of a conjunction *než* (than), first part

{This metarule handles a conjunction and a noun or a clause.}

```
A.syntcl = conj
A.compar = yes
```

{The item A is a comparative conjunction. The attribute compar is used instead of the attribute lexf in order to avoid problems with coding Czech characters (in the constraint A.lexf = než). The conjunction než is, at the moment, the only word in the dictionary having this attribute.}

```
IF B.syntcl = noun THEN
```

{The item B has syntactic properties of a noun. Since the comparative expressions such as *širší než delší* (wider than longer) are considered idiomatic, they should be handled by one of the phases of preprocessing, which is not yet implemented in the system.}

```
        B.case ? nom   Non_nominativ_compared
```

{The noun should be in the nominative case.}

```
ELSE    IF B.syntcl = v THEN ELSE
                IF B.syntcl = aux THEN ELSE FAIL ENDIF
        ENDIF
```

{The item B represents the governing word of the subordinate clause. It may be either the main (autosemantic) verb or the auxiliary verb.}

```
ENDIF
X := B
```

{The noun is a governing word.}

```
IF B.syntcl = noun THEN X.syntcl := compar ELSE
        X.syntcl := comparcl
ENDIF
```

{The new value of the attribute syntcl of the newly created item indicates that the item represents a comparative expression.}

```
OK
END_P
```

Metarule (35)

Comparison by means of a conjunction než (than) and the subordinate clause

{This is the second metarule from the set of metarules handling the comma and the conjunctive expression. The difference between this metarule and the following one consists in the presence of the comma. It is not allowed if the comparison does not involve a clause.}

```
A.syntcl = comma
```

{The item A represents a comma.}

```
IF B.syntcl = comparcl THEN
```

{If the item B represents the combination of a conjunction než and a subordinate clause (according to the previous metarule), then everything is O.K.}

```
ELSE    IF B.syntcl = compar THEN
                B.syntcl ? nonsens        Extra_comma_preceding_NEZ
```

{If the item B represents a noun combined with the conjunction než, it may not be preceded by a comma.}

```
        ELSE FAIL ENDIF
ENDIF
X := B
```

{The item B is a governing item of this metarule.}

```
X.syntcl := compar
```

{When the comma preceding the conjunction has already been processed the newly created item is assigned the same value of the attribute *syntcl* as in the case of the nominal comparison (metarule (34)). From the point of view of further processing both types of comparative expressions may be handled identically.}

```
OK
END_P
```

Metarule (36)

Comparison by means of a conjunction *než* (than) and the adjective in the comparative degree

{This is a final metarule of the set of metarules handling comparative expressions.}

```
IF A.syntcl = adj THEN
        A.deg ? comp    Wrong_grade
```

{The item A is an adjective in the comparative degree.}

```
IF B.syntcl = compar THEN
ELSE    IF B.syntcl = comparcl THEN
                B.syntcl ? nonsens        Missing_comma_preced_NEZ
```

{If the item B represents a subordinated clause and it has not been processed by the previous metarule (35), then there is a missing comma between the adjective and the conjunction *než*. Since the comma is obligatory in Czech in this case, the soft constraint marks a syntactic inconsistency.}

```
        ELSE FAIL ENDIF
ENDIF
X := A
```

{The governing item is the adjective.}

```
IF B.eos = yes THEN X.eos := yes ELSE ENDIF
IF B.end = yes THEN X.end := yes ELSE ENDIF
IF B.eoc = yes THEN X.eoc := yes ELSE ENDIF
```

{The information that the end of the clause (sentence) has already been processed is transferred further – cf. the comments on the metarule (3).}

```
OK
END_P
```

9.12 Pairs of expressions

This group of metarules handles expressions which may appear in pairs (*nejen – ale i*, *ani – ani*, *buď – anebo*, *jednak – jednak*, *jak – tak* etc.) [not only – but also; neither – nor; either – or, partly – partly, both – and]. They are usually classified as belonging to the class of coordinating conjunctions. The key to their processing is a special attribute *pair* containing two values (right, left). The "right half" is processed first and attached to the left expression. The left expression is attached to the rest of the sentence in a standard manner. The solution presented here is not ideal, it has to cope with natural restrictions of the approach to parsing used in the system (bottom–up propagation of data). It would be much more natural to use a certain kind of preprocessing module capable to locate and mark the presence of certain idiomatic pairs of expressions immediately after the morphological analysis is completed. Let us take the solution presented here as a clue supporting the argument that such a preprocessor could substantially improve the quality of the system.

Metarule (37)

The section following the right expression is being attached to the expression

```
A.pair = right
```

{The special attribute is tested. If the item A does not have this attribute, the processing of this metarule immediately stops.}

```
IF A.right = completed THEN
        A.right ? nonsens          Right_side_already_completed
ELSE    ENDIF
```

{This condition blocks all attempts to use this metarule several times for the same item.}

```
IF B.syntcl = noun THEN ELSE
    IF B.syntcl = v THEN ELSE
        IF B.syntcl = prephr THEN ELSE FAIL ENDIF
    ENDIF
ENDIF
```

{The item B may be either a nominal or a prepositional group or a clause.}

X := B

{Item B carries more information, therefore it was chosen as the governing item.}

X.pair := right

{The information that the resulting item is the right element of the pair is necessary for a proper attachment to the left element of the pair.}

X.right := completed

{This attribute blocks another application of this metarule to the item X later on.}

IF B.syntcl = v THEN ELSE X.nofm := yes ENDIF

{This condition blocks the attachment of a free modifier to the complete right section – any such attachment should be completed before this metarule is applied. This condition is, of course, valid only for a projective interpretation of this metarule.}

OK
END_P

Metarule (38)

Comma and the second half of the expression are being joined together

{This is the first of the pair of metarules handling the attachment of the right half to the left one.}

A.syntcl = comma

{The item A is a comma.}

```
IF B.pair = right THEN
    IF B.syntcl = conj THEN FAIL ELSE ENDIF
ELSE FAIL ENDIF
```

{The item B represents the complete right half of the expression. If B is a conjunction, it indicates that the previous metarule has not been yet applied.}

X := B

{The item B is the governing one.}

X.syntcl := rhalf

{New temporary value of the attribute syntcl is being introduced. It will be used in the next metarule.}

OK
END_P

Metarule (39)

The left and the right half of the expression are being joined together

{This is the second of the pair of metarules handling the attachment of the right half to the left one.}

A.pair = left

{The item A is an expression, which might become the left member of the pair.}

B.syntcl = rhalf

{The item B was derived as the result of the application of the previous metarule.}

X := A

{The item A is the governing one.}

X.pair := both
{The derived item receives a special value of the attribute *pair* marking the fact that the expression is complete.}
OK
END_P

9.13 Emphasis

Similarly as in the previous section the metarules introduced here handle a kind of a very specific construction. This construction, a special kind of apposition, typically emphasizes (repeats or specifies) a property of the governing word (cf. *Byl to hráč, a to hráč zatraceně dobrý!* (He was a gambler, namely a damned good gambler!))

Metarule (40)

{This metarule is the first member of a group handling the emphasis expressed by a coordinating conjunction and a demonstrative pronoun.}
A.syntcl = conj
A.conjtype = coord
{The item A represents the coordinating conjunction.}
B.prn_type = dem
{B stands for the demonstrative pronoun.}
X := B
{The pronoun is the governing item.}
X.syntcl:=stress
{Standard method of binding both metarules together through a unique value of the attribute syntcl.}
OK
END_P

Metarule (41)

{The second metarule combines comma and the item derived by the previous metarule.}
A.syntcl = comma
{The item A represents a comma.}
B.syntcl = stress
{The item B was derived by the previous metarule.}
X := B
{The item B is the governing one.}
X.syntcl := cstress
{The attribute syntcl receives the value necessary for the following metarule.}
OK
END_P

Metarule (42)

{The last member of this group of metarules attaches the emphasis to its governor (antecedent). Similarly as with free modifiers it is necessary to restrict the attachment to the first candidate preceding the emphasis.}

CLOSEST
{This keyword restricts the attachment to the nearest possible candidate.}
IF A.syntcl = v THEN ELSE

```
        IF A.syntcl = noun THEN ELSE
                IF A.syntcl = prepfr THEN ELSE FAIL ENDIF
        ENDIF
ENDIF
```

{The item A is either a verb or a noun (or a noun in the prepositional case).}

```
IF B.syntcl = cstress THEN ELSE
        IF B.syntcl = stress THEN B.syntcl ? cstress Missing_comma
        ELSE FAIL ENDIF
ENDIF
```

{If the item B was derived by the previous metarule, then the processing may continue. If it was derived by the metarule (40), it is necessary to point out the syntactic inconsistency (missing comma).}

```
X := A
```

{The item A should be the governing item.}

```
OK
END_P
```

9.14 Beginning and end of the sentence

The last pair of metarules handles the attachment of the final sentential marker and the left sentinel. The left sentinel serves as the root of the syntactic tree.

Metarule (43)

Attachment of the full stop, the question mark, the exclamation mark or the comma to the main verb of the clause (sentence).

```
PROJECTIVE
```

{This metarule may be applied only in a projective way.}

```
A.syntcl = v
```

{The item A represents a verb}

```
IF B.SYNTCL = int THEN
        ELSE IF B.SYNTCL = comma THEN ELSE FAIL ENDIF
ENDIF
```

{The item B is either a full stop, a question mark or an exclamation mark. It may also be a comma.}

```
X:=A
```

{The governing item is the item A.}

```
IF B.syntcl = int THEN X.eos := yes ELSE
        X.eoc := yes
ENDIF
X.end := yes
```

{It is necessary to store the information about the type of punctuation mark following the sentence (clause).}

```
OK
END_P
```

Metarule (44)

Left sentinel combined with the rest of the sentence.

```
A.wcl = sent
```

{The item A is the left sentinel.}

```
IF B.syntcl = v THEN
        IF B.eos = yes THEN ELSE FAIL ENDIF
ELSE   B.syntcl ? nonsens Sentence_not_completed
ENDIF
```

{If the item B does not contain information that the final punctuation mark has already been processed, it means that the sentence as not yet been properly parsed. The soft condition makes it possible to attach all isolated subtrees to the left sentinel in the negative parsing phases and thus to create at least a partial result even in case the grammar is not able for some reason to parse the sentence completely.}

```
X:=A
```

{The governing item is the item A – the left sentinel is thus a root of each syntactic tree.}

```
OK
END_P
```

9.15 General remarks

It is necessary to stress that our metagrammar does not aim at a complete coverage of the language, it rather contains a wide selection of phenomena illustrating that the task of building a metagrammar for a robust parser is very complex. It goes without question that our approach dividing the parsing process into several phases is a step in the right direction. The description of the metagrammar and, even more, the description of sentences from the testbed and discussion of results obtained in Chapter 11 show that we should pursue this strategy even further and to try to implement even more phases.

Each of these phases should exploit to the full extent the information available in the given moment and it also should concentrate on a single problem. Only the combination of methods may lead to a real success.

There is another important point. The experiments with the metagrammar described in this chapter helped both with the development of the software environment (and some of the grammatical problems encountered in this process influenced to a large extent the properties of the environment) and also stimulated the theoretical considerations introduced in previous chapters.

The experimental nature of the metagrammar is also one of the reasons why the parser is not yet connected directly to the module of the morphological analysis and the (complete) syntactic dictionary of Czech.

Chapter 10: Testing and debugging

One of the greatest problems of building a large scale non-deterministic grammar for automatic natural language parsing is the problem of adding new grammar rules (in the form of metarules representing sets of rules of the formal grammar) and improving or updating the old ones.

The same problem appeared also in our system. It was even more important to develop a method of testing and debugging the metagrammar because we have decided to build the metagrammar incrementally. It was no problem to keep track of changes when the grammar contained only metarules covering the most frequent syntactic phenomena of Czech, as for example the metarules for filling slots in valency frames or metarules dealing with prepositional, adjective or nominal groups. At the beginning of the process these metarules covered distinct areas of language, therefore there was no overlap between them. When other less frequent phenomena appeared among the sentences from the testbed, it was necessary to add new metarules and modify the existing ones.

These modifications represented in most cases a more subtle treatment of some phenomena. For example, when the problem of dealing with titles appeared, it turned out that the addition of a new metarule did not solve the problem, because in some cases it overlaped with the already existing metarules (3) and (5). That implied modifications of those metarules, namely an insertion of a block which more exactly defines the area of applicability of all three metarules within already existing metarules (3) and (5).

Similar problems appeared during the whole period of development of the metagrammar. For this reason it was necessary to develop a method of checking the consistency of the metagrammar in the process of testing and debugging.

The method used relies on a set of samples of input sentences (a testbed). The sentences contained in the testbed are divided into two groups – the first group consists of well-formed sentences, the second group contains ill-formed sentences. Each well-formed sentence from the first group has its counterpart in the second group, which is composed basically of the same words. The sentences in the testbed cover not only the syntactic phenomena which are already implemented in the grammar, but also some syntactic constructions which either were not yet implemented or which contain elements that cannot be captured by a metagrammar based solely on syntactic rules. The testbed thus demonstrates not only the phenomena which are parsable by our metagrammar, but also points out the limits of the approach we have chosen. The sentences were chosen for the testbed on the basis of a very simple rule – each new pair of sentences (the well-formed and the ill-formed version) added into the testbed should contain at least one interesting phenomenon which was not present in the testbed until then. Some of the sentences are described in a more detail in the next chapter.

The interpreter of the grammar was implemented by Tomáš Holan primarily for the LATESLAV project [Holan et al. 97] and it was later modified according to the requirements of our robust parser. It has a wide variety of switches allowing to influence its behavior and to debug the grammar more easily. For example, one of the switches (t) makes it possible to minimize the size of the output by suppressing the messages about the application of individual metarules. The output then looks as follows (the sentence *Tehdy jsem byl ministrem financí.* – Then I was a minister of finances.):

```
-------------------------
LEFT_SENTINEL:1..1
-------------------------
TEHDY:2..2
-------------------------
JSEM:3..32
-------------------------
BYL:33..92
```

```
--------------------------
MINISTREM:93..93
--------------------------
FINANCÍ:94..95
--------------------------
".:96..96
======= CYK parser ========

#
#
# TEHDY JSEM BYL MINISTREM FINANCÍ ".
# sample11.dat           ==>>       0.66s OK    2 (Best)  trees
Minimal trees = 2
      0.66
```

The first part of the output contains the information about individual input words and about the number of unambiguous items created from each word. The last four lines then contain the information about the input sentence and about the results of parsing, namely the name of the file containing the input sentence, the processing time, the actual result of parsing and the number of syntactic trees created. The quotation mark preceding the full stop in the input sentence was inserted for purely technical reasons – the full stop, similarly as some other characters (i.e. question mark, comma, exclamation mark, the characters a, b, x, p, A, B, X and P etc.), is reserved and may not be used for any other purpose.

The shortened form of the output of all sentences from the testbed is combined in one file and everything except the text and the actual results of the parsing of all sentences is filtered out. The file then looks as follows (this sample of the file contains the results of the first four sentences of the testbed):

```
      #
      #
      # JAK JE TO S KAŽDOU TOUTO NĚJAKOU JEHO VLASTNÍ SPORTOVNÍ ČINNOSTÍ "?
      # sample1.dat                 ==>>       0.72s OK    1 (Best)  trees
      #
      #
      # KDS NEPŘEDPOKLÁDÁ SPOLUPRÁCI SE STRANOU PANA SLÁDKA "A NENÍ PRAVDOU
", ŽE PŘEDSEDA KŘESŤANSKÝCH DEMOKRATŮ PAN BENDA PROSADIL V TELEFONICKÉM
ROZHOVORU S PETREM PITHARTEM ING. DEJMALA DO FUNKCE MINISTRA ŽIVOTNÍHO
PROSTŘEDÍ ".
      # sample2.dat                 ==>>      4.12s OK    16 (Best)  trees
      #
      #
      # KAŽDÝ MÁ HÁJIT PRÁVO ", PŘEDEVŠÍM PRÁVO OBČANA NA OCHRANU CTI A
DŮSTOJNOSTI ", NA KTERÉ MÁ KAŽDÝ OBČAN NÁROK PODLE LISTINY ZÁKLADNÍCH PRÁV
A SVOBOD ".
      # sample3.dat                 ==>>       2.08s OK    20 (Best)
trees
      #
      #
      # HORNICKÁ ZDRAVOTNÍ ZAMĚSTNANECKÁ POJIŠŤOVNA ( HZZP ) SE VČERA
ODVOLALA PROTI ROZSUDKU KRAJSKÉHO SOUDU ", KTERÝ VYHLÁSIL NAD TOUTO
POJIŠŤOVNOU KONKURSNÍ ŘÍZENÍ ".
      # sample4.dat                 ==>>       0.60s OK    2 (Best)  trees
```

The listing contains several characters preceded by quotation marks (“?, “A, “,, “.). All these symbols are reserved characters used in grammar metarules (cf. Chapter 7), the quotation marks are necessary in order to distinguish normal input characters from those used in metarules as special reserved symbols.

The file containing the listing, called SAMPLE, is then manually compared with the master file, called MASTER, which contains the same type of results, the only difference being the fact that the file MASTER had been created before the changes in the grammar were made. Because an important role in the comparison is attributed to the processing time (a sudden substantial increase of processing time may indicate serious troubles), it is necessary to perform all tests on the same computer. All tests presented here were carried on a Mobile Pentium III 600MHz with 128MB of memory, running Windows 98 operating system.

During the manual comparison of both files it is necessary to concentrate on the number of best trees and also on the processing time. If either of these figures changes, it is necessary to parse a particular sentence once more in order to receive the full listing of results. In a standard setting (no switches applied) the interpreter provides the parsing information in the following form:

```
97/97      3:6 [8]     2..3    C(B)
98/98      3:6 [8]     2..3    C(B)
99/99      3:7 [8]     2..3    C(B)
100/100      3:8 [8]     2..3    C(B)
…
714/1      1:666 [49]    1..7    A(D(C(B)E(F)G))
715/1      1:690 [49]    1..7    A(D(C(B)EFG))
716/1      1:698 [49]    1..7    A(D(BCE(F)G))
717/1      1:712 [49]    1..7    A(D(BCEFG))
```

The sample of the result shows the first and the last four lines of data describing the application of individual metarules. Each line represents an application of one metarule. The data in each line are organized in the following way:

a/b   *c:d [e]*   *f..g*   *h*

where

a is the number of the item created during the application of a rule *e* on already existing items number *c* and *d*.

b is the number of the physical slot where the item *a* is stored.

f is the number of the leftmost word (the input words are numbered from left to right) and *g* is the number of the rightmost word of the interval of words covered by the item *a* (in case of discontinuous coverage in nonprojective phases numbers of all words covered by the item *a* are listed here instead of the interval.

h is a linearized syntactic tree corresponding to the item *a*. The content of each pair of brackets represents a list of daughters of the node preceding the left bracket. The letters correspond to the numbers of words, for example A represents the word No. 1, B is the word No. 2 etc.

The parsing information provided by the interpreter in this form is usually sufficient for the localization of any problem which might be caused by the modification of the grammar. If this information is not sufficient, it is possible to run the interpretation of a sentence once more with a switch **?** which makes it possible` to display individual items accompanied by the information about numbers of their parents and the number of the rule according to which the given item was created.

When all the problems encountered during the comparison of MASTER and SAMPLE files are solved, the SAMPLE is taken as a new MASTER. This procedure is repeated after every update of the grammar or after every addition of a new sentence into the testbed. In this way it is possible to maintain the system simply and efficiently in a consistent way.

Chapter 11: The testbed

As has been already mentioned in previous chapters, a well-balanced testbed is a very important part of the system, which may save a lot of troubles during the process of testing and debugging the system. In this chapter we would like to present and comment on a set of sentences which at the moment serve as the testbed for the system of a robust parser of Czech. Let us take the sentences one by one discussing the syntactic phenomena they contain. All sentences have a counterpart – an ill-formed sentence containing identical or nearly identical words. All input sentences contain also a symbol of the left sentinel, the leftmost input item indicating the beginning of each sentence. In the syntactic representation the left sentinel always serves as the root of the syntactic tree.

All data about processing times of particular sentences are measured on the computer with Mobile Pentium III processor running at 600 MHz with 128MB of memory. The interpreter of the grammar runs under the MS-DOS operating system in the minimal output mode, the results are sent to a file and are not displayed on the screen (these specifications are important for the processing time listed by each metarule, if the results are displayed on the screen, the parsing is slower). The trees representing ill-formed sentences display in the black and white mode negative symbols as dashed lines starting in the node with the negative symbol and going down towards the dependent node involved in the application of the rule that created the negative node. The alternative way of displaying the results on the computer screen uses on the black background red lines instead of dashed ones, while the lines representing the positive applications of rules are yellow. The figures used in this chapter always show an actual screenshot of a result in a black and white mode. In this way some of the trees, especially those belonging to longer sentences, may be slightly unreadable in some places due to the fact that the placement of nodes in the tree is done automatically and the program uses only very simple rules for the tree design. In case that the system provides more than two results we list only two of them as an illustration. In such a case we always try to list the results which are more interesting than the others from some point of view.

Sentence (1)

Jak je to s každou touto nějakou jeho vlastní sportovní činností?

[Lit.: How is it with every this some his own sporting activity?]

This is an artificially created sentence which was designed with the purpose of testing the complex rules for the attachment of a congruent adjectival attribute from the left-hand side. The sentence is well formed even though it is slightly unnatural. The parser provides one result (Fig.11.1.).

The form of the tree is influenced by the use of the keyword CLOSEST in the metarule dealing with the attachment of free modifiers. Would that keyword not be present, the result would contain one more tree in which the subtree headed by the noun *činnosti* (activity) would depend directly on the verb *je* (is). This situation is quite typical not only for Czech – the decision about the governing node of a free modifier is performed with respect to the meaning of the sentence; from the purely syntactic point of view it is always ambiguous. Such an ambiguity is, of course, very expensive from the computational point of view since as we can see e.g. in the second sentence, a combination of more free modifiers in one clause has a disastrous effect on the number of results even with this keyword applied, the more so without it.

The chosen restriction on the attachment of free modifiers is regular in the sense that it is always attached to the closest possible node in the tree, but it may depend also on all verbal (infinitive)

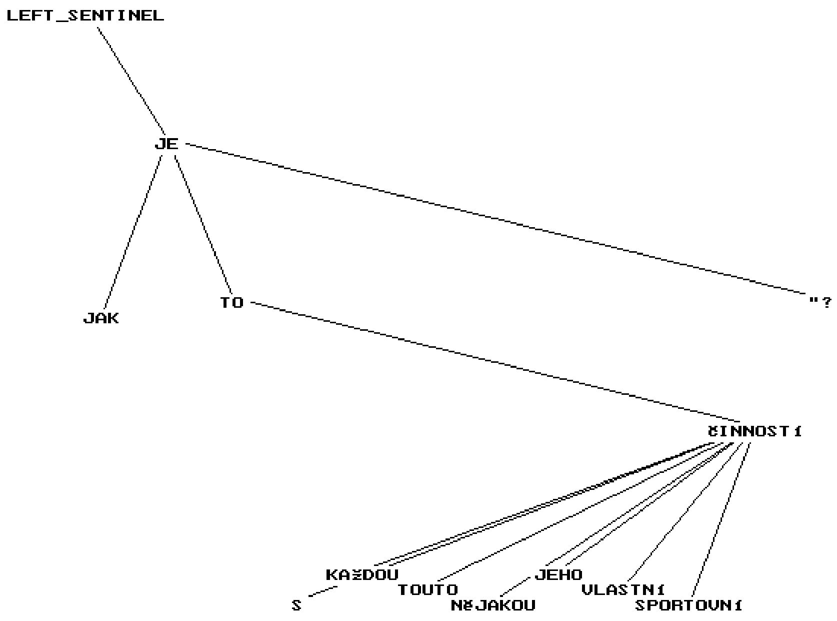

Fig. 11.1. The tree representing a syntactically correct variant of the sentence (1)

and nominal nodes on the branch which is incidental with the main verb and ends in the node governing the free modifier. This regularity makes it possible to reconstruct all readings if needed. The processing time was 0,72s. There were 280 items created during the parsing process.

An ill-formed counterpart of this sentence is the sentence:

Jak je to s každou touto nějakou jeho jakoukoliv vlastní sportovní činností?

[Word for word translation: How is it with every this some his any own sporting activity?]

The parser provides two results. The first one contains four syntactic inconsistencies (negative symbols) and after a comparison with the second syntactic tree (one syntactic inconsistency) it can be filtered out without doubts. The syntactic inconsistency contained in the second tree indicates that the word *nějakou* (some) can not depend on the noun *činností* (activity), because there is a word *jakoukoliv* (any) already depending on the same noun and it is not possible to combine two indefinite pronouns as attributes of the same noun.

The processing time was 0,98s. There were 708 items created during the parsing process.

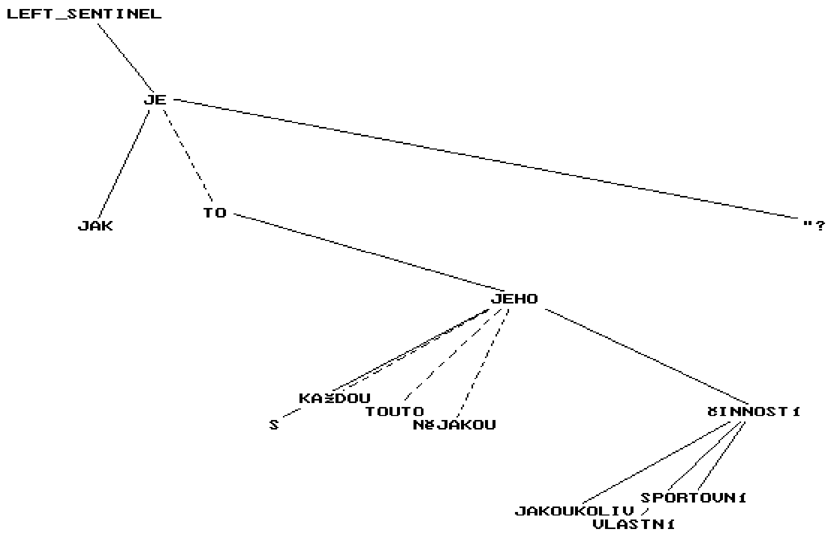

Fig. 11.2. One of the trees representing the syntactically ill-formed variant of the sentence (1)

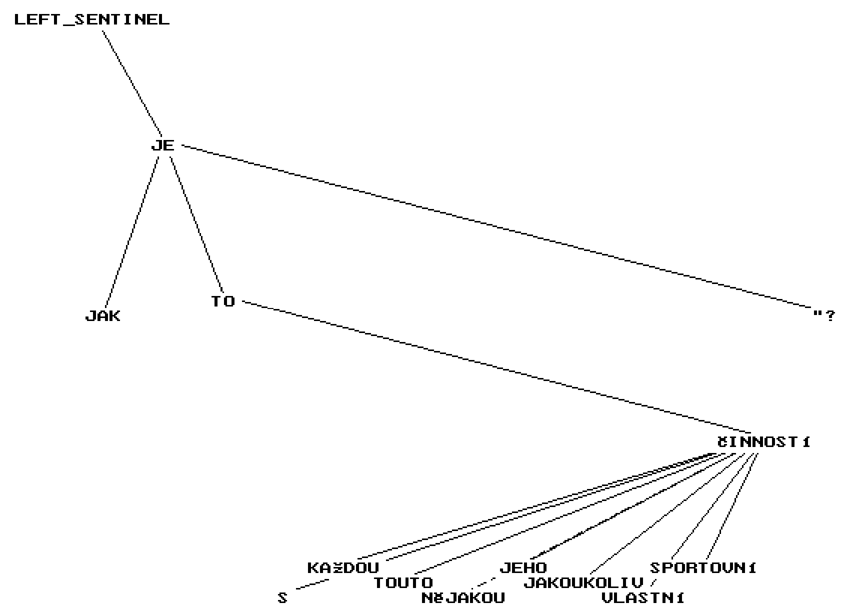

Fig. 11.3. The second tree representing the syntactically ill-formed variant of the sentence one (the syntactic inconsistency is marked between the words *nějakou* and *činností*)

Sentence (2)

KDS nepředpokládá spolupráci se stranou pana Sládka a není pravdou, že předseda křesťanských demokratů pan Benda prosadil v telefonickém rozhovoru s Petrem Pithartem ing. Dejmala do funkce ministra životního prostředí.

[Lit.: CDP [does] not_suppose cooperation with party [of] Mister Sládek and [it] isn't true that chairman [of] Christian Democrats Mister Benda promoted in phone discussion with Petr Pithart ing. Dejmal to function [of] minister [of] environment]

(The CDP does not suppose the cooperation with the party of Mister Sládek and it is not true that the chairman of Christian Democrats Mister Benda was promoting in phone discussion with Petr Pithart engineer Dejmal into the position of the minister of environment.)

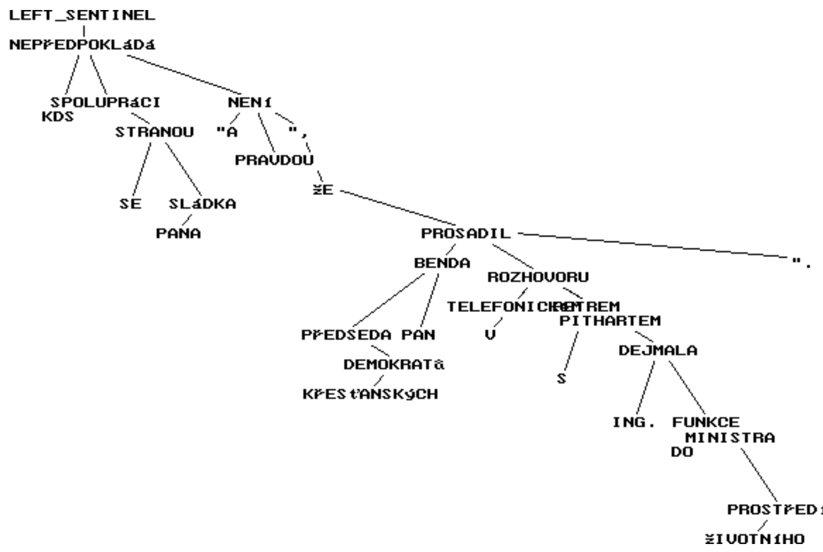

Fig. 11.4. One of the trees representing the syntactically correct variant of the sentence (2)

This sentence is an unchanged newspaper sentence (from Lidové noviny). It was included into the testbed for the following reasons:

- It allows to test how the system will react to long sentences.

- It contains a number of words which serve as a title and therefore allows to test the interaction between the metarule (4) and other metarules.
- It is a good illustration of difficulties connected with the presence of a higher number of free modifiers in one clause,
- It is complex and therefore provides a good ground for testing a wide variety of metarules in addition to those already mentioned above.

The system provides 16 syntactic trees, the differences between them are concentrated to the lower right corner of screenshots, the main difference being in the attachment of free modifiers and the existence of two candidates for an object of *prosadil* [promoted], namely the words *Dejmala* and *ministra* [minister (acc.)]. Even though the restriction on the attachment of free modifiers to the immediately preceding antecedent is applied, the number of possible combinations is quite high.

The processing time was 4,12s. There were 2330 items created during the parsing process.

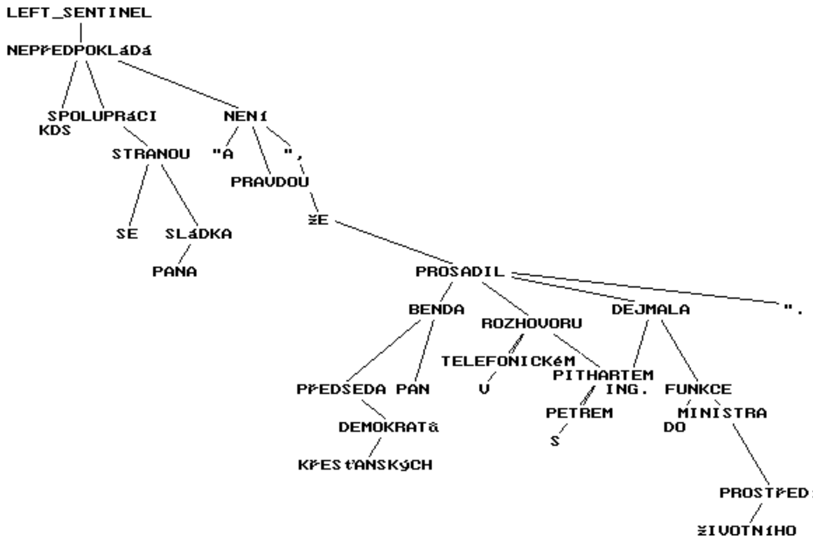

Fig. 11.5. Another tree representing the syntactically correct variant of the sentence (2)

An ill-formed variant of this sentence lacks the comma preceding the subordinate conjunction *že* (that). According to the research carried in the frame of the LATESLAV project [Petkevič 95] punctuation errors are among the most frequent types of errors in Czech. The majority of native speakers are not sure about the proper placement of commas in more complicated sentences. Many of these errors are automatically undistinguishable, on the other hand some types of errors are easily recognizable and can be handled automatically even in long and complicated sentences.

The missing comma preceding *že* belongs to the latter group due to the fact that in Czech the isolated subordinated conjunction *že* should *always* be preceded by a comma. Only in case that the subordinated conjunction is accompanied (preceded) for example by an adverb, pronoun or another conjunction (*a* [and]) the situation is not so straightforward.

The claim that this type of syntactic errors is not complicated is supported by the number of derived items – 2599 is only slightly more than in the previous case – and by the number of trees – 16 – the same number as in the previous case. On the other hand, the processing time is almost twice as long – 7,25s. This disproportion is caused by the pruning mechanism – the number of derived items reflects only those items which were not discarded during the parsing process immediately after they were created (mostly because "better" items had already been created before). The process of creation of these items takes time.

The graph representing one of the 16 trees shows that the system is really capable to locate a syntactic inconsistency and to create proper syntactic trees corresponding to the trees representing the syntactically correct variant of this sentence. For the purpose of the demonstration of variants of an attachment of noun phrases other than in the two trees above the following tree was selected:

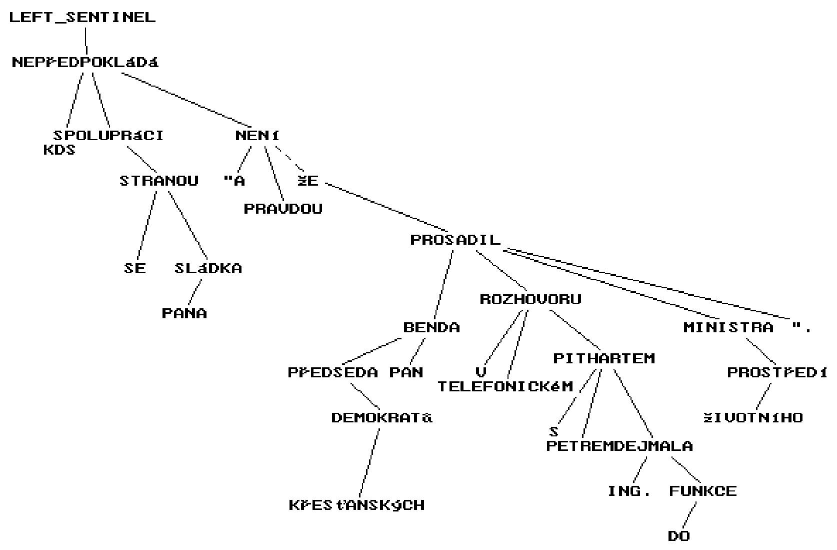

Fig. 11.6. One of the trees representing the syntactically ill-formed variant of the sentence (2)

Sentence (3)

Každý má hájit právo, především právo občana na ochranu cti a důstojnosti, na které má každý občan nárok podle listiny základních práv a svobod.

[Lit.: Everyone should defend right, above_all right [of] citizen on protection [of] honor and dignity, on which has every citizen claim according_to declaration [of] basic rights and liberties.]

(Everyone should defend rights, above all the citizen's right of protection of his honor and dignity, which is granted to every citizen according to the declaration of basic rights and liberties.)

This is again a newspaper sentence. It contains four new phenomena, not contained in the previous two sentences:

- A modal verb followed by a verb in the infinitive form
- An inserted parenthetical noun phrase "above all the citizen's right of protection of his honor and dignity"
- Two examples of nominal coordination – "honor and dignity" and "rights and liberties"
- A subordinate relative clause containing the relative pronoun *který* (which) in the prepositional case as a connector between the main and the subordinated clause

The problem of modal verbs can be handled by standard methods – the valency frame of the modal verb contains slots for subject (noun in the nominative case) and object (verb in the infinite form). The metarule for filling the valency slots (the metarule (5)) does not make it possible to fill the subject slot in the valency frame of the verb in the infinitive form. This simple mechanism makes it possible to handle even more complicated structures as for example: *Petr měl chtít začít pracovat dříve*. [Lit.: Petr should to_want to_begin to_work earlier.] (Petr should have wanted to start working earlier.).

The inserted phrases and sentences (typically in apposition to a word from the main sentence) can in fact be handled separately from the main sentence, the sentence itself does not change its syntactic structure if we omit the inserted phrase. A substantial problem of inserted phrases is to find what belongs to the phrase and what does not. We have to take into account that in the moment of processing the items (words) belonging to the inserted noun phrase there is no information that the particular items actually are parts of an inserted noun phrase. That does not matter for example in attaching attributes or free modifiers, but the decision whether the final comma is really the *final* character of the insertion cannot be done at the moment when the comma is processed. We can easily modify the sentence for example as follows:

Každý má hájit právo, především právo občana na ochranu cti a důstojnosti, ***svobody projevu a pohybu a jiných základních lidských práv****, na které má každý občan nárok podle listiny základních práv a svobod.*

[Everyone should defend rights, above all the citizen's right of protection of his honor and dignity, **the freedom of speech and movement and other basic human rights***,* which is granted to every citizen according to the declaration of basic rights and liberties.]

The modified sentence shows that even the inserted phrase may have quite a complicated syntax and that it is really impossible to decide on the basis of the local context whether the comma following the word *dignity* is the final comma of the inserted phrase or not. For such a decision it is necessary to take into account a broader context. The following chapter introduces a mechanism, which might help to solve this problem on the basis of identification of syntactically unambiguous items delimiting individual segments of a sentence and building a graph roughly describing the structure of clauses in complex sentences.

The problem of coordination is also very complex and it is again very difficult to determine the proper syntactic representation on the basis of local context only even if we take into account only the projective readings. The sample syntactic trees of this sentence show that even the simple nominal coordination *Listiny základních práv a svobod* ([of the] Declaration of basic rights and liberties) has three parses. Two of them are more or less obvious, in one of them the adjective *základní* [basic] modifies both nouns, in the other it modifies only the closest noun. The third reading sounds strange to most human readers, it is marked in the relevant tree by an inserted rectangular frame (Fig. 11.9). The coordination concerns the nouns *listiny* [document, genitive sing.] and *svobod* [freedoms, genitive pl.]. These readings may be described by means of the following set of DR-trees:

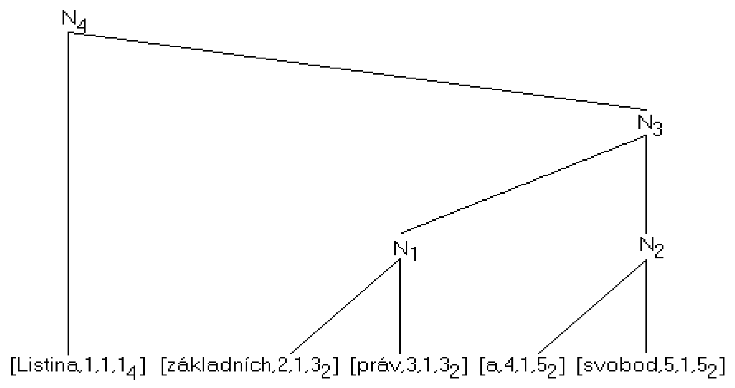
N_4
N_3
N_1
N_2
[Listina,1,1,1_4] [základních,2,1,3_2] [práv,3,1,3_2] [a,4,1,5_2] [svobod,5,1,5_2]

Fig. 11.7a The adjective modifies the closest noun

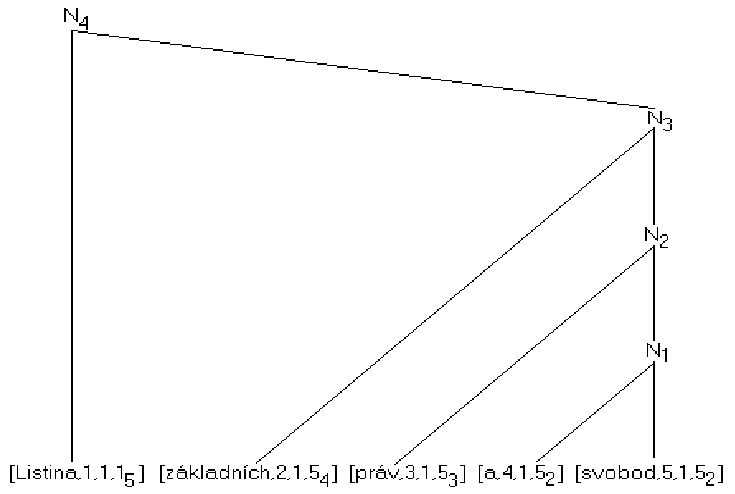
N_4
N_3
N_2
N_1
[Listina,1,1,1_5] [základních,2,1,5_4] [práv,3,1,5_3] [a,4,1,5_2] [svobod,5,1,5_2]

Fig. 11.7b The adjective modifies the whole coordination

The symbols N_i in Fig.11.7a stand for: $N_1=[T_1, 3, 2,5_3]$, $N_2=[T_2, 5, 2,5_3]$, $N_3=[T_3, 5, 3,1_4]$, $N_4=[T_4, 1, 4, 0]$. In all nodes of all three DR-trees from Fig. 11.7a-c the symbols T_i stand for relevant nonterminal symbols actually used in the grammar. We do not use actual nonterminal symbols here due to the considerably higher complexity of grammar rules necessary for a proper derivation of these DR-trees compared for example to the grammar used in the Example 5.1.

The symbols N_i in Fig.11.7b stand for: $N_1=[T_1, 5, 2,5_3]$, $N_2=[T_2, 5, 3,5_4]$, $N_3=[T_3, 5, 4,1_5]$, $N_4=[T_4, 1, 5, 0]$.

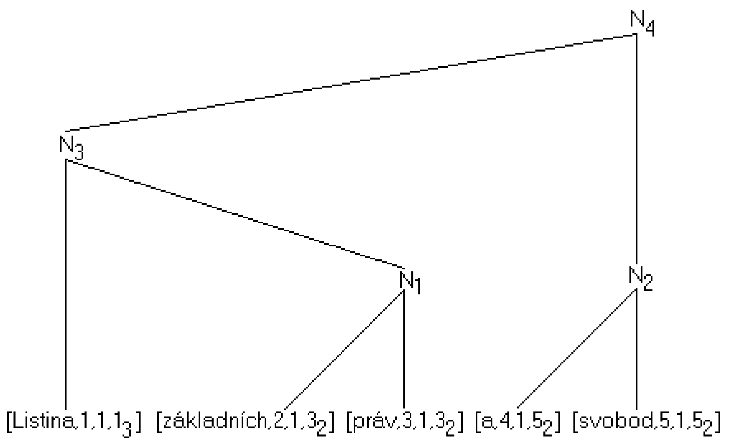

Fig. 11.7c The adjective modifies the closest noun and the coordination concerns the nouns *Listiny* [document] and *svobod* [freedoms].

The symbols N_i in Fig.11.7c stand for: $N_1=[T_1, 3, 2,1_3]$, $N_2=[T_2, 5, 2,5_4]$, $N_3=[T_3, 1, 3,5_4]$, $N_4=[T_4, 5, 4, 0]$.

Even more complicated is the coordination *cti a důstojnosti* (honor and dignity). The complicated structure of the inserted nominal group *především právo občana na ochranu cti a důstojnosti* also allows multiple variants of coordination – on the basis of purely syntactic rules the noun *důstojnosti* (dignity) may be coordinated also with nouns *ochranu*, *občana* and *právo* (protection, citizen and right, respectively). Such a high number of variants is responsible for the high number of results obtained – 20.

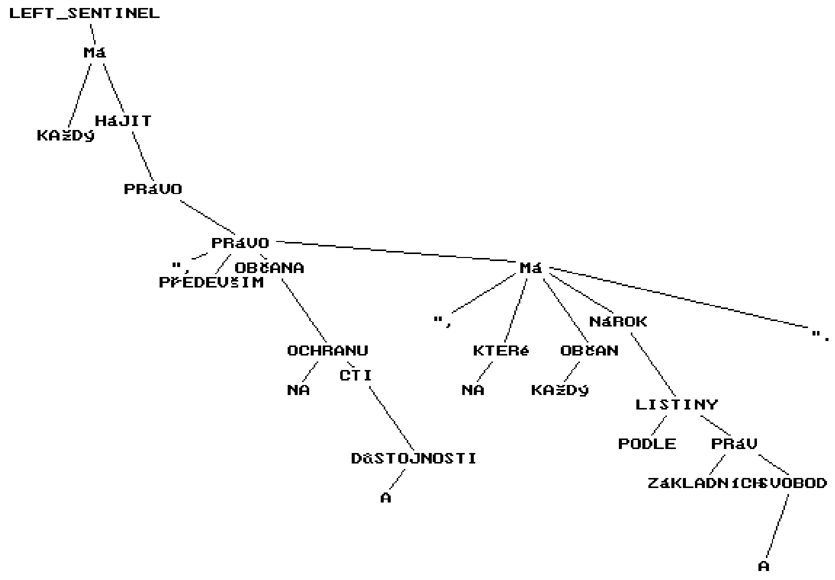

Fig. 11.8. One of the trees representing the syntactically correct variant of the sentence (3)

It is quite clear that this situation is very similar to the attachment of free modifiers in a sense that these variants differ only locally, they are beyond the scope of the pure surface syntax and they burden the parsing process with a high number of unnecessary items, all of which may be safely generated later (from some kind of "canonical representation" of the coordination, describing probably the coordination of the closest pair of candidates). The problem of coordination thus calls for an analogous solution (the reduction of the number of parses by means of a keyword CLOSEST in the relevant metarule).

When we have applied the keyword CLOSEST to the metarule handling coordination of nouns in the process of testing and debugging our metagrammar, we have got only 2 syntactic trees and 525 derived items. That represents a substantial difference against 20 trees and 1269 derived items obtained without the application of the keyword CLOSEST to the nominal coordination. The reason why we have finally decided not to use the keyword is our feeling that it would need a detailed linguistic observation to be sure that this manner of handling coordination is adequate and that there is no danger of losing important information in the process of analysis.

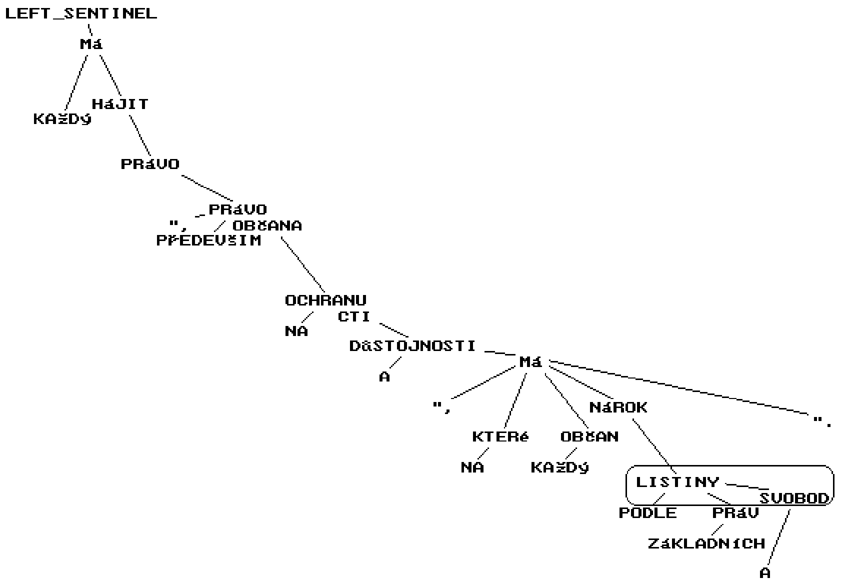

Fig. 11.9. Second tree representing the syntactically correct variant of the sentence (3)

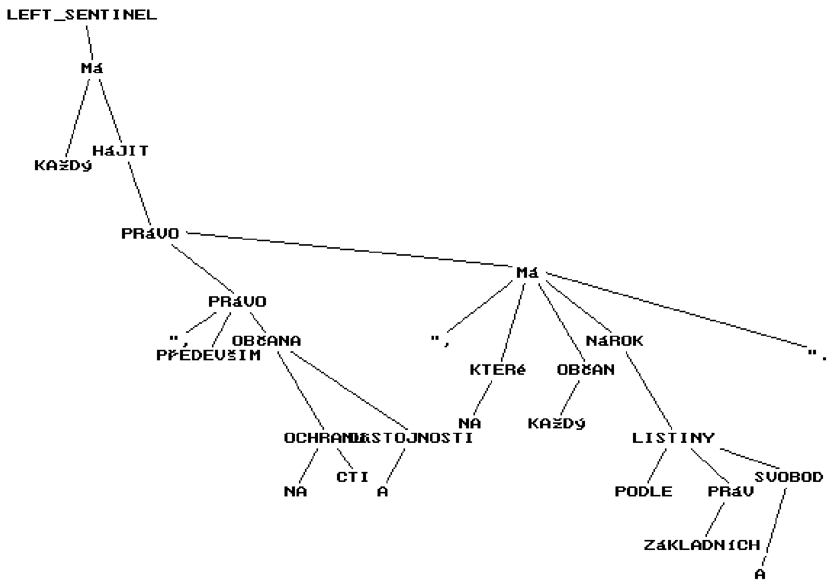

Fig. 11.10. The third tree representing the syntactically correct variant of the sentence (3)

The relative clauses are simpler from the point of view of the treatment in this system, although in general it is the other way round. The agreement in gender and number of the antecedent and the pronoun provides enough information for automatic processing. The problem is that in case there are more competing antecedents (five in this case – twice *právo*, then *ochranu*, *cti* and *důstojnosti*) the decision which possible antecedent the relative pronoun refers to is usually semantically based. The sample syntactic trees of the sentence in Fig. 11.8 and 11.9 show the ambiguity of the relative clause attachment. The edge between words *OCHRANU* and *CTI* is not displayed in Fig.11.9, it is overwritten by the latter word. This sometimes happens due to the automatic way of preparing the layout of the trees and the simplicity of the relevant subroutine.

The combined ambiguities of coordination and attachment of the relative clause are responsible for the number of syntactic trees of the sentence. The processing time was 2,08s.

There is one more interesting feature in this sentence – from the syntactic point of view it is also possible to attach the relative clause to the noun *právo* (right) from the main sentence (cf. Fig.11.10). The reason for this attachment is the fact that the comma following the parenthetical inserted complex nominal group has two roles. It not only divides the inserted nominal group and the relative clause depending on the noun *právo* (right) from the inserted group, but it also serves as a final comma of the inserted nominal group. This is a very good illustration of the complexity of the problem of "virtual commas", the commas which are represented by a single comma even though they have more than one syntactic role (in the sense of formal syntax, not in the sense of the natural language syntax – single comma in fact functions instead of several commas).

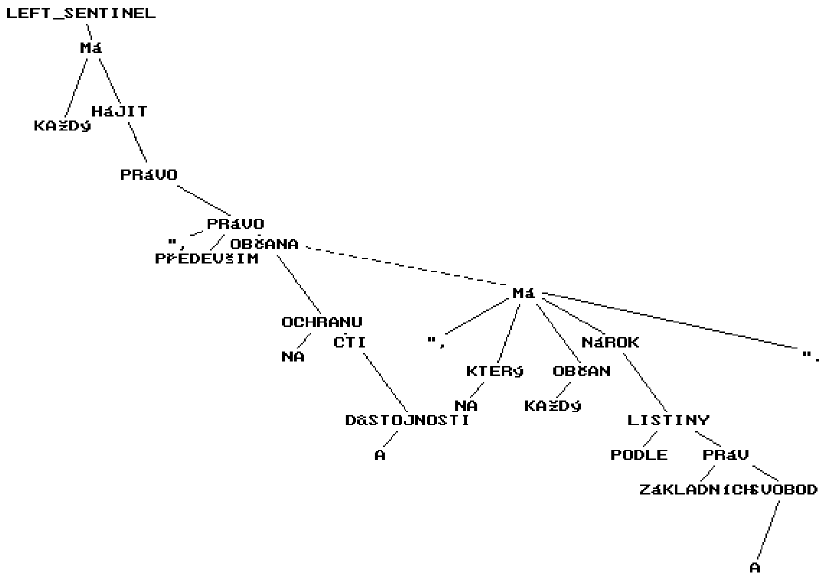

Fig.11.11. One of the trees representing the syntactically ill-formed variant of the sentence (3)

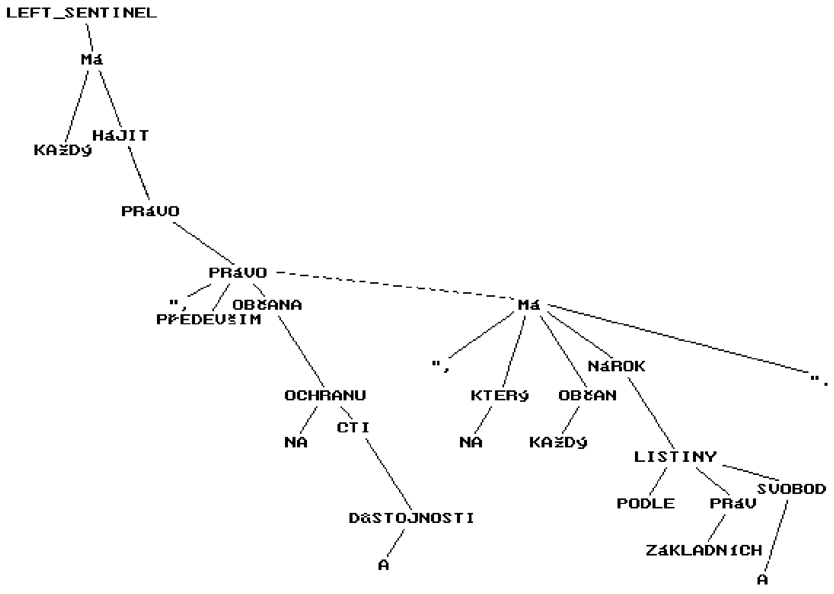

Fig. 11.12. Second tree representing the syntactically ill-formed variant of the sentence (3)

The ill-formed variant of the sentence differs from the original one only in the form of the relative pronoun *které*, which is substituted by an inappropriate form *který*. The problem is that for this form there is no possible antecedent. The robust parser is able to locate the inconsistency and to reconstruct all possible variants of the attachment of the relative clause, as shown on the trees representing the sentence in 11.11 and 11.12. The greater number of possible antecedents than in the previous case leads to 104 parses obtained after 9,29s, with 2889 derived items.

Sentence (4)

Hornická zdravotní zaměstnanecká pojišťovna (HZZP) se včera odvolala proti rozsudku krajského soudu, který vyhlásil nad touto pojišťovnou konkursní řízení.

[Miner health employee's insurance (HZZP) itself yesterday appealed against verdict [of] regional court, which declared over this insurance bankrupt trial.]

(The Employee's health insurance company of miners (EHICM) appealed yesterday against the verdict of the regional court which had declared over this insurance company a trial of bankruptcy.)

The interesting phenomena contained in this sentence are the following:

- Parentheses
- Reflexive verb
- The presence of an adverb modifying a verb
- Relative clause containing the relative pronoun in a nonprepositional case

The treatment of parentheses is similar to the treatment of inserted phrases with one substantial difference – the right parenthesis is unambiguous and thus the parsing of the content of parentheses does not create so many structures as in the case of inserted clauses. Reflexive verbs are treated by the grammar only to a certain extent – only the actual presence of a reflexive particle is checked, not its position. The treatment of positions is not, according to our strategy, a task to be handled by the metagrammar, it is a matter of application of global constraints imposed on structures created during the parsing process. This treatment is identical e.g. to the treatment of nonprojective constructions, as has already been mentioned in previous chapters.

The adverbs are parsed according to the value of the attribute *addep* indicating the kind of the word the adverb depends on.

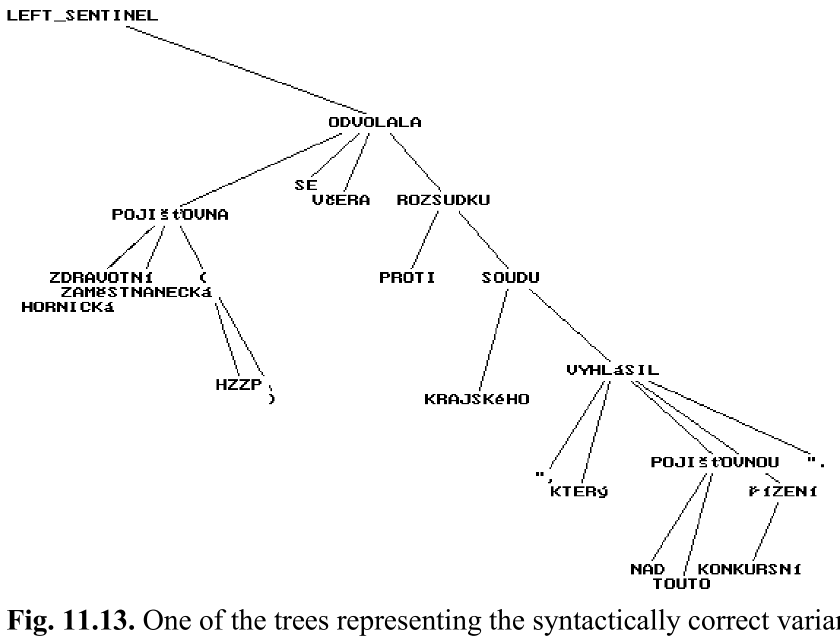

Fig. 11.13. One of the trees representing the syntactically correct variant of the sentence (4)

The treatment of relative clauses containing the relative pronoun in a nonprepositional case is similar to the treatment of sentences with the pronoun in a prepositional case with one exception – the relative pronoun in a nonprepositional case usually represents an inner participant of the main verb of the subordinate clause. Let us remind here that we have simplified valency frames of verbs in our metagrammar to the extent that they do not contain any inner participants in prepositional cases (cf. the discussion in Chapter 8.4). For this reason it is necessary to check whether a relative pronoun in nonprepositional case can fill one of the slots in the verbal valency frame of the main verb of a particular relative clause.

As shown in the figures of syntactic trees of the sentence, another important problem of parsing relative clauses is the difficulty to find the antecedent of the relative pronoun. When no semantic filters are applied, all possible candidates must be taken into account. This might be another area, where the use of the keyword CLOSEST would substantially reduce the number of parsing results. We have decided not to use it in metarules taking care about relative pronouns, because we feel that a detailed linguistic investigation should also be done for this phenomenon before any simplifications can be applied.

The parser provides 2 syntactic trees, the parsing takes 0,60s and 446 items are derived.

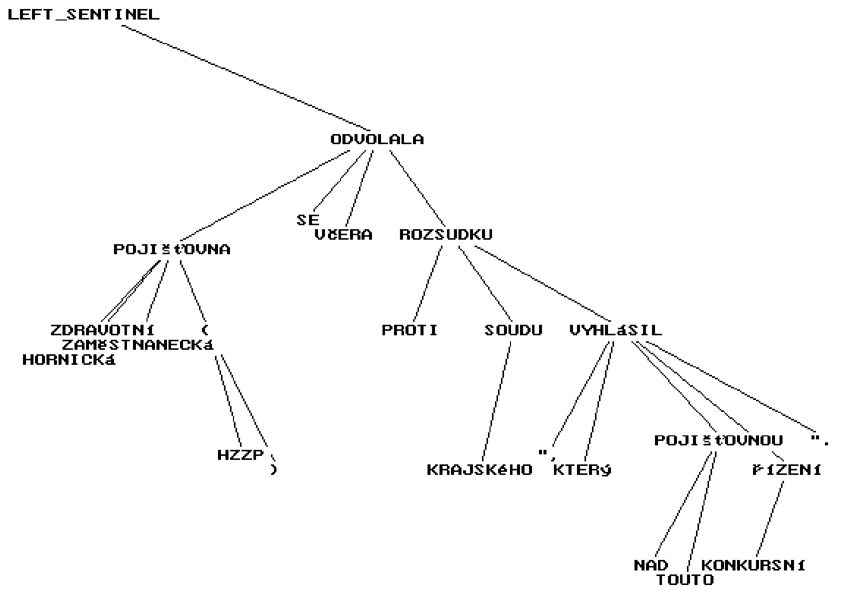

Fig. 11.14. Second tree representing the syntactically correct variant of the sentence (4)

A variant of this sentence was created by removing the reflexive particle *se* from the original sentence (*pojišťovna odvolala* instead of *pojišťovna se odvolala*). Since the verb *odvolat* is not a reflexive tantum, the parser accepts the sentence as syntactically correct and creates its syntactic trees almost identical to those belonging to the original variant of the sentence. This result might be a topic for further discussion, because this sentence is in fact ill-formed, due to the missing obligatory direct object. Even though this situation is not so regular and frequent in Czech as an omission of the subject of the sentence, the robust parser in general cannot dare to mark the sentence with the missing direct object as syntactically ill-formed. There are several examples of sentences where the obligatory inner participants are not present in sentences even though they were not explicitly mentioned in the previous context, mainly in those sentences where the obligatory inner participant is a part of the "common knowledge" and as such it mustn't be explicitly mentioned (Cf. the sentence *Galileo nakonec odvolal*. (Galileo finally recanted.)). On the other hand, our system allows a very simple treatment of sentences with too many non-prepositional participants – once a slot is filled, it is removed from the frame and cannot be filled again.

At this point it is important to mention one very important fact. Even though the system does not consider sentences with missing obligatory inner participants as being ill-formed, the fact that an inner participant is missing is marked inside the syntactic structure. As we have already mentioned in the previous paragraph, whenever a slot of the frame is filled, it is removed. All slots remaining in the frames after the analysis is completed, represent missing participants. This information can be easily retrieved from syntactic trees. Let us remind that the obligatory inner participants are really obligatory on the tectogrammatical level, not on the surface level, where the presence (or absence) of inner participants is substantially influenced by the context. Due to our general strategy which considers individual sentences (not paragraph or documents) to be main input units it is impossible to open the problem of incorrectly omitted inner participants.

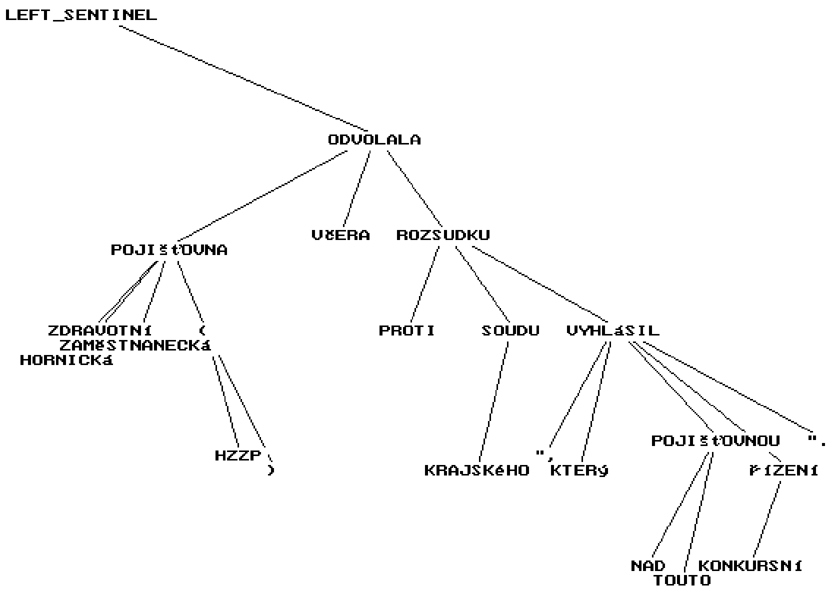

Fig. 11.15. One of the trees representing the modified variant of the sentence (4)

The parsing of the modified sentence takes 0,50s, 467 items are derived and 2 trees are created.

Sentence (5)

Tento úřad by podle jeho návrhu neměl podléhat vládě ani být orgánem výkonné moci.

[This office Cond. according_to his proposal shouldn't be_liable to_government nor to_be body of_executive power]

(According to his proposal, this office should not be liable to the government nor to be a body of executive power.)

There are three reasons why this sentence is included into the testbed:

- Conditional form of the verb
- Coordination expressed by means of the conjunction *ani* [*nor*]
- The ambiguous word form *moci* [to be able/power(genitive, dative or locative case)]

The first phenomenon does not constitute a difficult problem from the point of view of automatic parsing. The auxiliary verb in the conditional form behaves to a great extent similarly to the reflexive particle *se*. Its treatment is therefore similar. It is necessary to ensure that only one auxiliary may be connected to a particular main (meaningful) verb and that the main (autosemantic) verb is in the past participle. The conditional form of verbs is processed by the metarule (7) (and its counterpart dealing with the reversed order of main and auxiliary verbs) described in section 9.3.

The second phenomenon is more complicated. In this sentence the conjunction *ani* [nor] is used as a gradation coordination where comma is not used, while in conjunctive coordination (where a pair *ani* – *ani* usually occurs – cf. sentence (13)) the comma before the latter *ani* is obligatory. The main problem is that the conjunction *ani* (nor) may be generally used both in the conjunctive and in the adversative. If it is used in the conjunctive, there is no comma preceding this conjunction, while in adversative the presence of a comma preceding the conjunction is necessary. The difference between these two cases is mostly purely semantic, therefore it is outside the scope of our robust parser. There are no means how to distinguish syntactically in which manner the conjunction *ani* is used.

The last problem is the ambiguous word *moci*. It illustrates very well the danger of using the parameter CLOSEST in metarules. As is shown in Fig. 11.16 and Fig. 11.17, one of the differences responsible for the number of syntactic structures we obtain as the result of parsing this sentence is the attachment of the final period to one of the verbs present in the sentence.

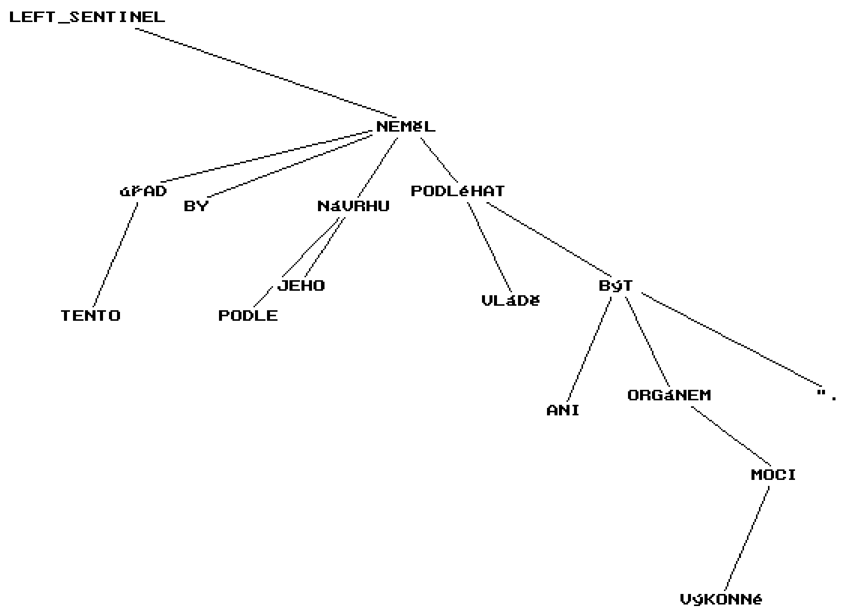

Fig. 11.16. One of the trees representing the sentence (5)

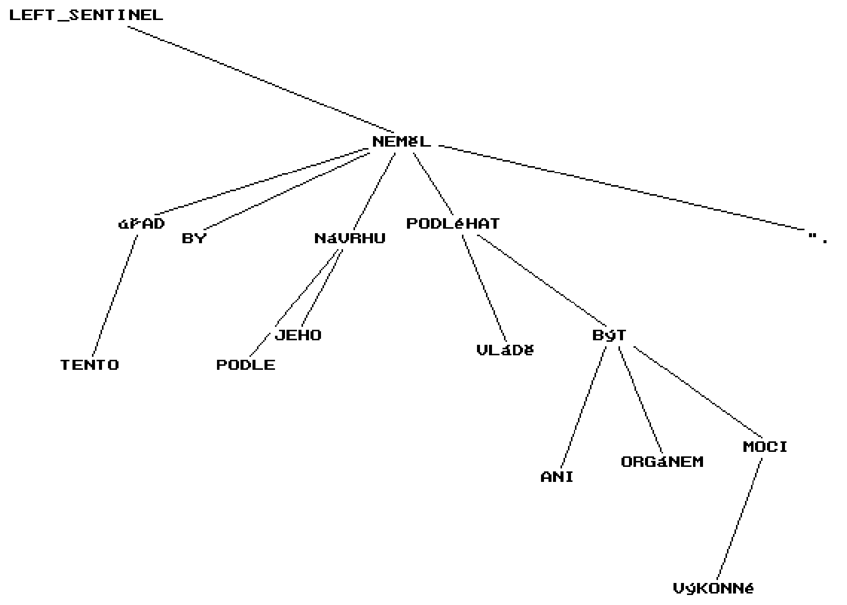

Fig. 11.17. Another tree representing the sentence (5)

It would be natural to attach the period to the closest verb and thus to reduce the number of trees from 6 to 2, but unfortunately it can not be done. The problem is the ambiguity of the word form *moci*, which is both an infinite modal verb [to be able] and the form of the noun *moc* [power, might] in several cases of both singular and plural (gen., dat., voc. and loc sg; nom., acc. and voc. pl.) The word form is then represented by eight unambiguous items, one representing the verb and seven describing a noun in one of the applicable combinations of cases and numbers. The verbal item then blocks the attachment of the period to any other verb in the sentence – it is the closest one to the period and the system is not able to take into account that there are alternative nominal items that are used in the "right" syntactic trees. This is to a certain extent a weak point of the software used for the interpretation of the metagrammar – once the unambiguous items are created, they become independent and there is not a way how to get any information about their "brothers". The fact that a modal verb cannot be combined with the auxiliary verb *být* (to be) in the order *být moci* (the other way round, *moci být* is possible, cf. *moci být dobrý* (to be able to be good)) in one clause does not help, because at the moment when the final period is being attached to the modal verb, no other context is being taken into account (due to the nature of the interpreter, which does not make it possible to work with any other item than just the two which appear on the left-hand side of a particular metarule).

The problem of the ambiguity of the form *moci* clearly indicates that a really reliable disambiguation of input word forms might help tremendously even in case that it would not be able to disambiguate all ambiguous word forms. The stochastic disambiguation of Czech (cf. [Hajič, Hladká 98]) is not yet able (and it is really questionable whether it ever will) to work reliably enough for our needs.

There are of course ways of overcoming the weaknesses of the interpreter even without the application of any kind of morphological disambiguation (at least in some specific cases). For example, we may assign a special attribute, called *AlternativePOS,* to each item representing a word form which is assigned more than one part of speech, even before the unambiguous items are created. In this manner the author of the metagrammar will be able to formulate constraints and conditions taking into account also the fact whether there is an alternative for a part of speech of a given (unambiguous) item or not.

This solution may be useful in some cases, but it would not solve the problem of the attachment of the period to the closest verbal item. The information about the existence of a "related" item with different part of speech is not sufficient here. Even if we had an access to this information, we can not forbid the period being attached to the verb with an alternative part of speech. Such a simple solution would not work in sentences containing *only* ambiguous verbs. If every verb in the sentence will have also an alternative part of speech, the system would never attach the final period to any of the verbs (cf. the sentence *Pastevci ženou*[verb/noun] *stáda na pastvu*. (The shepherds drive the herds to the pasture.)). We need the information whether the verbal item was actually used in any syntactic structure covering the whole sentence, i.e. whether there is at least one plausible reading of the sentence with the relevant item being a verb. That is unfortunately far too complicated information.

It is quite clear that on the level of the metagrammar it is very difficult to find a flawless solution. It would be much better to aim at a definition of some kind of general constraint determining the candidates for period attachment. Such a constraint could take into account a broader context and thus it also could avoid the trap of the restrictions of the context in metarules.

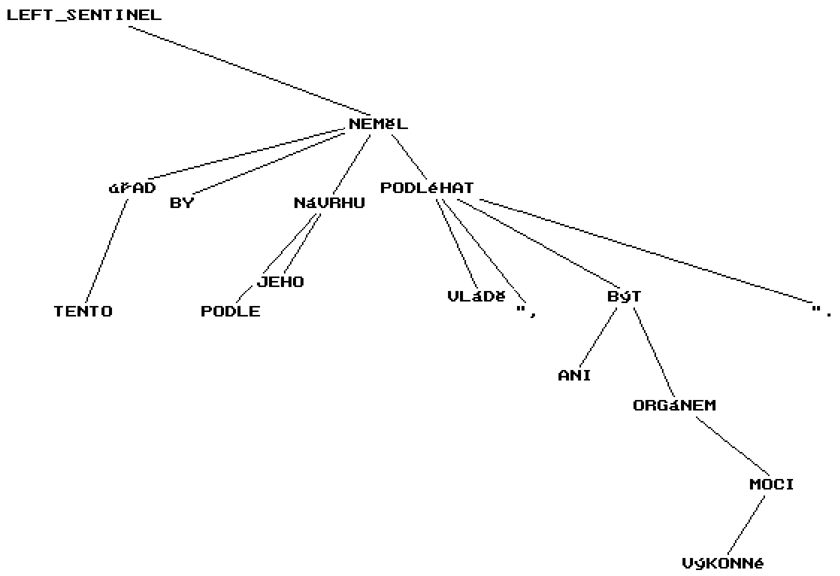

Fig. 11.18 One of the trees representing the variant of the sentence (5)

The above mentioned facts have led us to the decision not to use the keyword CLOSEST in the metarule handling the attachment of the final period to any suitable verb in the sentence and thus we have obtained 6 syntactic trees (differing in the attachment of the period and also in the attachment of the nominal group *výkonné moci*, which may either be an incongruent attribute of the noun *orgánem* or an inner participant of the verb *být*. The sentence was parsed in 0,28s and 383 items were created in the process.

The second variant of the sentence contains a comma preceding the conjunction *ani* (nor). As we have already mentioned, the presence of comma in this sentence is a matter of understanding or interpretation of the sentence (the difference between adversative and conjunctive use of the conjunction *ani* is purely semantical), therefore our robust parser accepts it as syntactically well-formed, too, and we get also 6 trees as in the previous case. The presence of the comma slightly increases both the number of items derived (438) and the processing time 0,44s.

Sentence (6)

Ministerstvo zdravotnictví jako její zřizovatel by mělo během víkendu rozhodnout o případné urychlené likvidaci této jedné z největších zdravotních pojišťoven u nás.

[Ministry [of] healthcare as her establisher should have in_the_course_of weekend to_decide about contingent accelerated liquidation [of] this one from biggest health insurance_companies at us.]

(The ministry of healthcare as its establisher should decide in the course of the weekend about the contingent accelerated liquidation of one of the biggest health insurance companies in our country.)

This sentence contains further interesting phenomena we have not met yet:

- A complement *ministerstvo jako zřizovatel,* (ministry as establisher) which is syntactically indistinguishable from the comparative construction of the type *pes jako slon* (a dog like an elephant)
- Selective construction *jedné z největších* (one of the biggest)

The comparative construction represents a large group of phenomena, which must be treated by special metarules applicable to individual lexical entries or small groups of words. The only problem is to restrict the application of these metarules only to the relevant group of words. Sometimes it is difficult to draw a borderline between constructions which should be handled by a full-scale parser and idioms which are simple enough to be parsed by means of some kind of a preprocessing automaton.

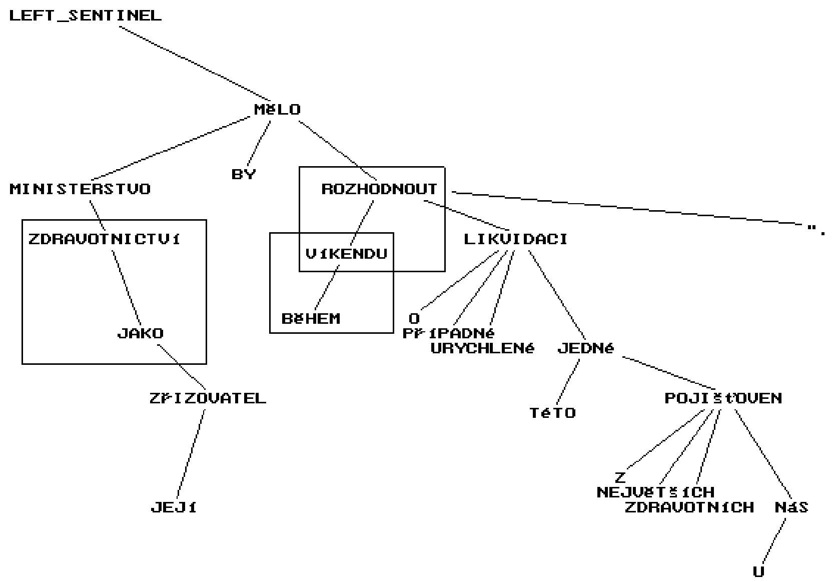

Fig. 11.19. One of the trees representing the syntactically correct variant of the sentence (6)

Although the selective construction looks at first sight similar to the comparative construction or to the complement, it is different. The problem is the agreement in gender of the numeral and the head of the prepositional phrase (cf. *jedné*[fem.] *z největších pojišťoven[fem]* – **jednoho*[anim., inan., neut.] *z největších pojišťoven*) very often the agreement is not a pure syntactic phenomenon, but it is semantically based (cf. *jeden*[inan.,anim.] *ze třídy* (one from the class) – *jedna*[fem.] *ze třídy*). This distinction may not be captured by a surface syntactic parser. In order to avoid the problem of marking the syntactically correct sentences as ill-formed (in case the agreement is not required) our parser does not treat the selective construction of this type differently from the attachment of ordinary prepositional phrases and no agreement is checked here.

There are 12 parses with 515 derived items. The processing takes 0,50s. The areas marked on the displayed syntactic trees show that the number of parses is caused by the ambiguity of attachment of the comparative construction and by the ambiguity of the word *během*[prep./noun] (in the course of/by running). If the word *během* is taken as a secondary preposition, then the prepositional phrase *během víkendu* may be attached to both verbs. There is also the same problem of period attachment present in this sentence as in the previous one.

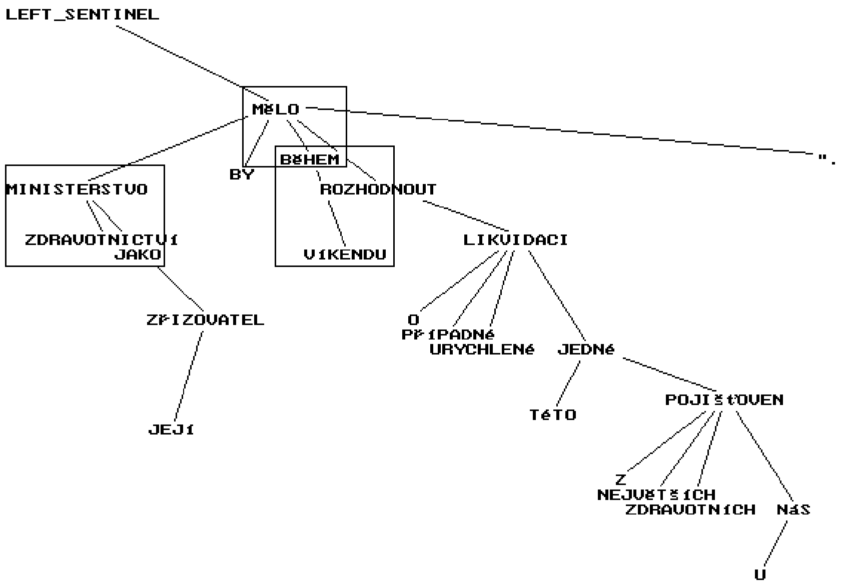

Fig. 11.20. Second tree representing the syntactically correct variant of the sentence (6)

A modified variant of the sentence contains one error which is quite common in texts written by the help of word processors – the conditional form of the auxiliary verb *by* is incorrectly used twice. This type of errors (missing or redundant words) is caused by changes made by the author of the text when (s)he copies or deletes whole parts of sentences.

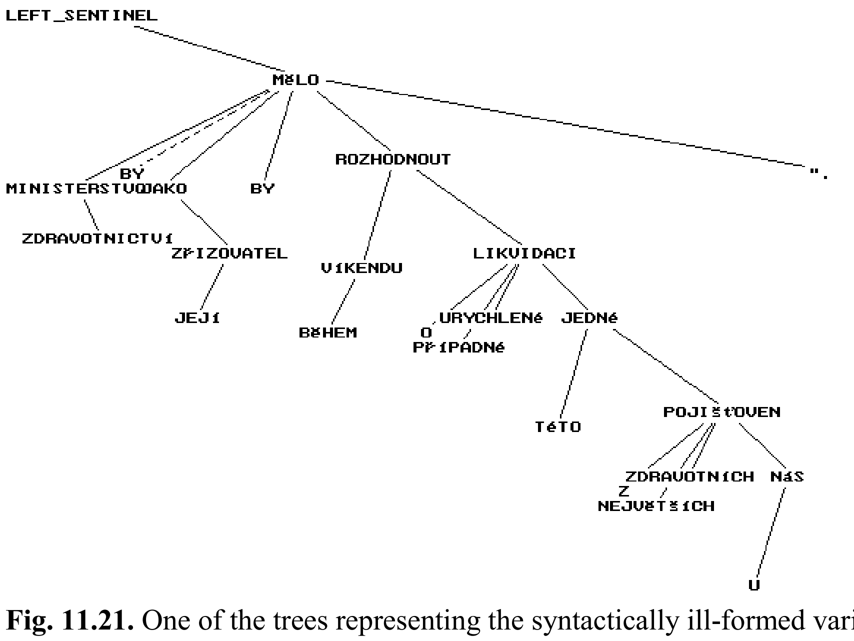

Fig. 11.21. One of the trees representing the syntactically ill-formed variant of the sentence (6)

The modified sentence then looks like this:

Ministerstvo zdravotnictví **by** jako její zřizovatel **by** mělo během víkendu rozhodnout o případné urychlené likvidaci této jedné z největších zdravotních pojišťoven u nás.

[Ministry [of] healthcare **should** as her establisher **should** have in_the_course_of weekend to_decide about contingent accelerated liquidation [of] this one from biggest health insurance_companies at us.]

(The ministry of healthcare ***should*** as its establisher ***should*** decide in the course of the weekend about the contingent accelerated liquidation of one of the biggest health insurance companies in our country.)

The tree representing the syntactic structure of the ill-formed variant of the sentence is one of the six results the parser issues. The smaller number of trees is caused by the fact that the inserted auxiliary verb *by* blocks the attachment of the comparative construction to the nouns *ministerstvo* and *zdravotnictví* in the projective phase. The parsing process took 1,16s and 844 items were derived.

Sentence (7)

Na první pohled hrůzostrašný sjezd není nebezpečnější než jiné lyžařské disciplíny.

[On first sight horrific downhill_run isn't more_dangerous than other ski disciplines.]

(The downhill run, horrific at first sight, is not more dangerous than other ski disciplines.)

This sentence contains a different type of comparative construction than the previous sentence – second degree (comparative) of an adjective combined with the conjunction *než* (than) and a nominal group. The sentence is quite simple, it has only one syntactic tree. The processing time is 0,50s and 307 items were created.

It might be argued that the attachment of the nominal group *Na první pohled* (at first sight) should be attached rather to the immediately following adjective *hrůzostrašný* (horrific) than to the verb (cf. Fig.11.22). Let us therefore repeat at this point that for the reason of reduction of the number of parses the nominal groups in prepositional cases are being attached solely to verbs and nouns due to the high degree of ambiguity of free modifier attachment. Any correction of the syntactic tree based on the actual content of valency frames of words from the sentence and on the result of parsing by our parser is possible later, in a special module or phase following the robust parser. For a very detailed discussion of problems of attachment of nominal groups in prepositional cases see [Straňáková 01].

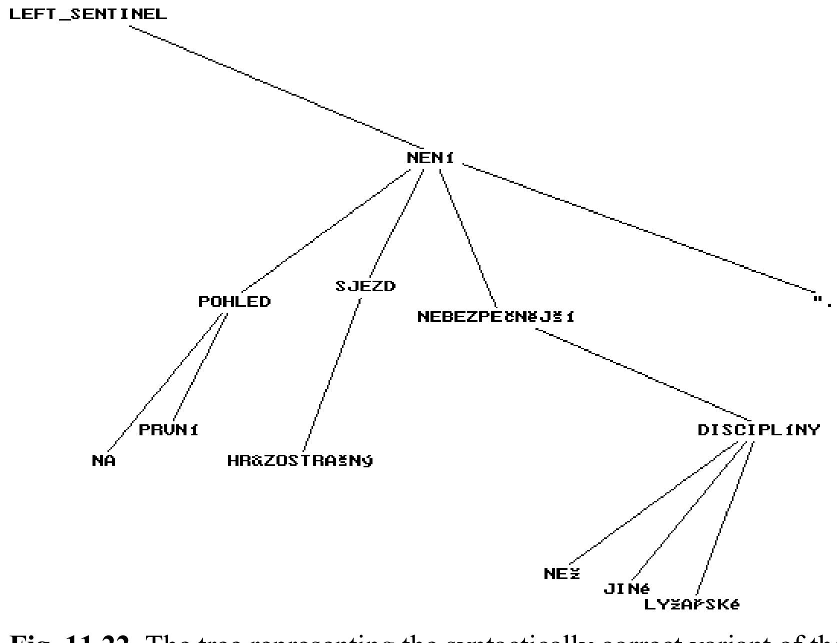

Fig. 11.22. The tree representing the syntactically correct variant of the sentence (7)

The second remark concerns the comparative construction containing the word *než.* This conjunction may appear also in different types of constructions, not only in the type presented in this sentence.

Another possibility of using and analyzing the construction containing this conjunction is being discussed in the comments concerning the sentence (14).

The ill-formed variant of the sentence contains the adjective *nebezpečný* (dangerous) instead of *nebezpečnější* (more dangerous). The system is able to identify the syntactic inconsistency (there is neither the comparative adjective nor a verb present in the clause – cf. the sentence (14)) and it creates an unambiguous syntactic representation of the sentence corresponding to the structure of the original sentence. The processing takes 0,39s and 332 items were created.

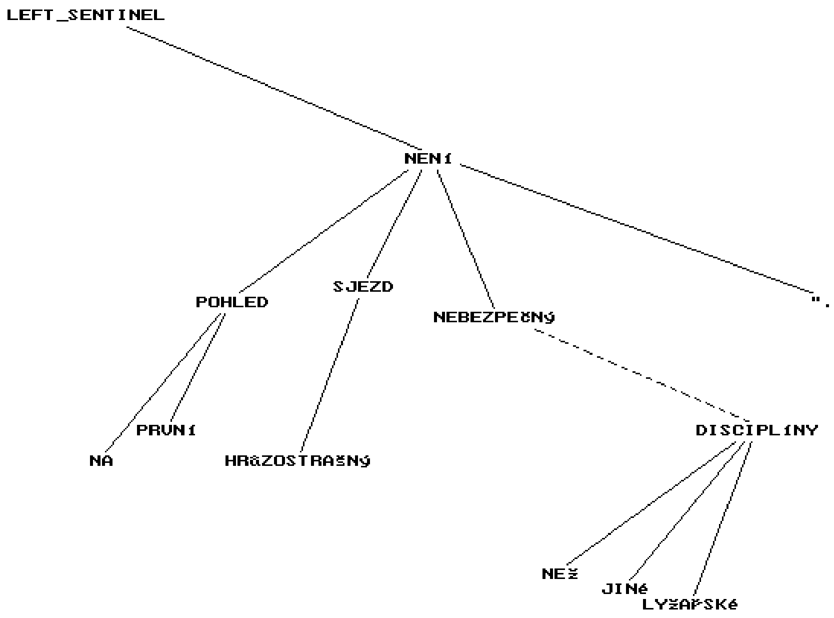

Fig. 11.23. The tree representing the syntactically ill-formed variant of the sentence (7)

Sentence (8)

Tehdy jsem byl ministrem financí.

[Then I_am was minister [of] finances.]

(Then I was a minister of finances.)

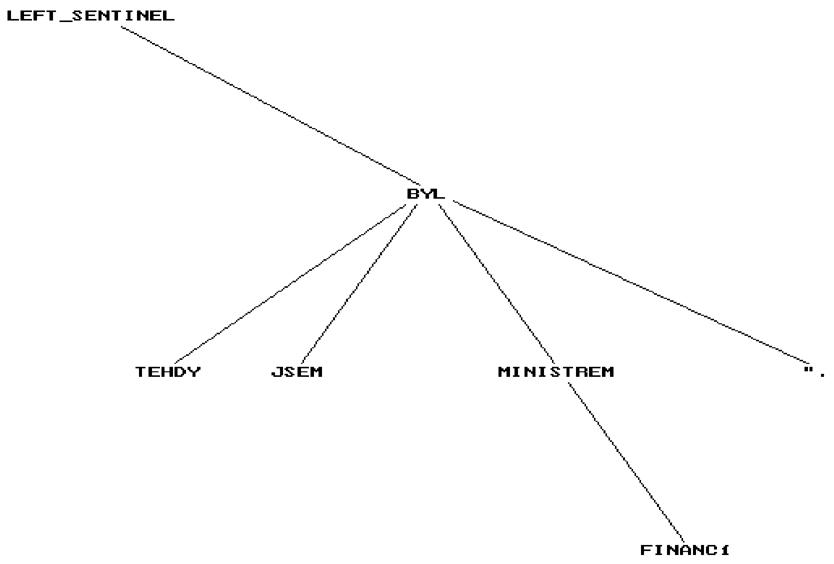

Fig. 11.24. The first tree representing the syntactically correct variant of the sentence (8)

Even though this newspaper sentence is very short, it contains two interesting phenomena:

- Past participle in the 1st person sg.
- Nominal predicate in instrumental case

The main problem of this sentence is the combination of the past participle and the nominal predicate – this means that the predicate is in fact composed of three individual words, two of them being auxiliaries derived from the same basic form *být* (to be).

The metagrammar provides two syntactic structures for this sentence. The trees differ in the attachment of the noun *ministrem*. Fig. 11.24 shows this word as a part of the nominal predicate, the Fig. 11.25 on the other hand shows this word as a free modifier (nouns in the instrumental case often have in Czech a role of the free modifier), while the word *financí* is attached as a nominal predicate. Both trees are created after 0,66s. 382 items are derived during the parsing process.

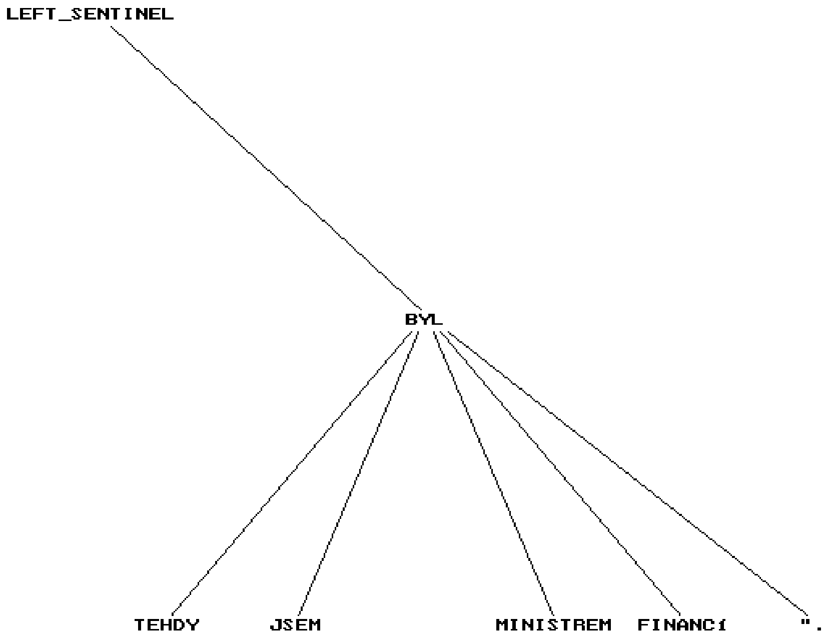

Fig. 11.25. The second tree representing the syntactically correct variant of the sentence (8)

Modification:

Tehdy jsme byl ministrem financí.

[Then we_are was minister [of] finances.]

(Then we was a minister of finances.)

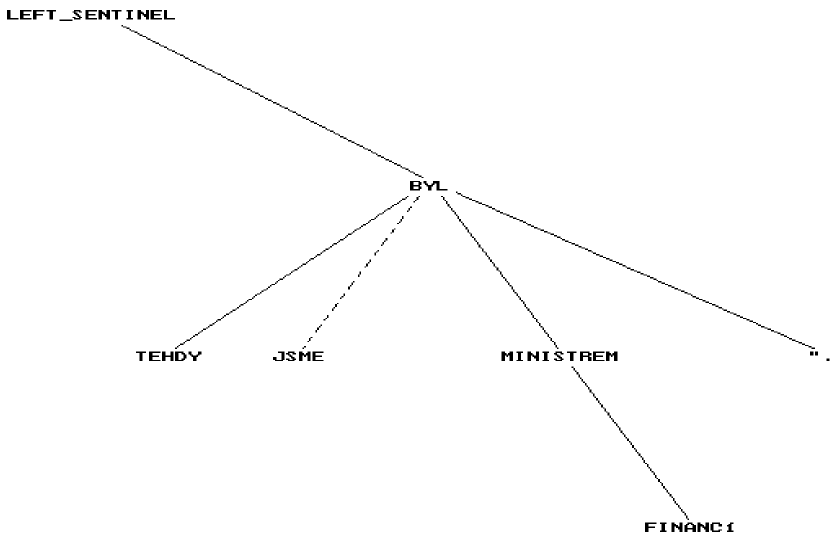

Fig. 11.26. The first tree representing the syntactically ill-formed variant of the sentence (8)

The ill-formed variant of the sentence contains one very common error undistinguishable by a spelling checker. It is a quite common mistake to invert the order of characters *e* and *m* when fast typing the auxiliary verb *jsem* [1st pers.sg] and to type *jsme* [1st pers.pl] instead (or vice versa). Since both forms exist in the language, the spell checkers are not able to mark any of them as erroneous. The syntactic analysis of the sentence results in the right solution – the disagreement in number between *jsme* [pl.] and *byl* [sg.] is marked in both syntactic trees representing the sentence. Both trees are otherwise identical to the syntactic trees representing the correct variant of the sentence.

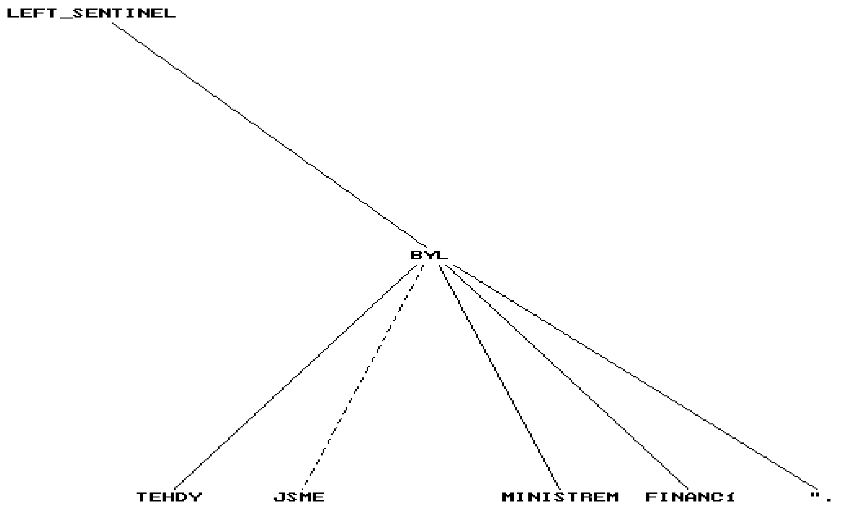

Fig. 11.27. The first tree representing the syntactically ill-formed variant of the sentence (8)

The processing of the ill-formed variant took 1,49s and 818 items were derived.

Sentence (9)

Za nejlepší způsob, jak dosáhnout trvalého rozvoje Slovenska, považuje SNS i nadále samostatnou Slovenskou republiku..

[For best way, how to_achieve permanent development [of] Slovakia, considers SNP [Slovak National Party] hereafter independent Slovak republic.]

(As a best way of achieving a permanent development of Slovakia SNP further considers an independent Slovak republic.)

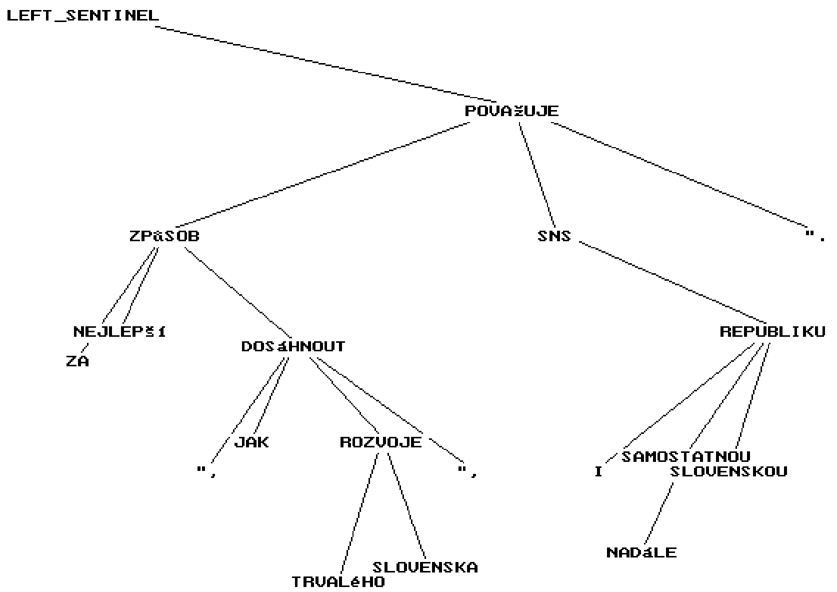

Fig. 11.28. First tree representing the syntactically correct variant of the sentence (9)

This sentence contains one new phenomenon, the inserted clause in the role of an attribute of the immediately preceding noun. There is one basic difference between this sentence and the sentence three – in this case the inserted part is not only a noun phrase, but it includes a verb. The inserted part is connected to the governing noun by means of a pronominal adverb *jak* (how). No agreement is required. The treatment of this construction is thus very similar to the treatment of inserted noun phrases, the difference being only in the type of the head of the inserted clause. The relevant metarule handling this construction contains a special constraint blocking its application to those types of inserted relative clauses where the agreement is required in order to avoid the situation that the inserted relative clause is not processed by a "proper" metarule (because there is a disagreement between the relative pronoun and the governing noun), but it is incorrectly processed by the metarule handling inserted clauses in general.

The figures represent two variants of syntactic trees out of the three results obtained. The processing took 0, 27s and 265 items were derived.

The result shown in Fig. 11.28 is interesting, it reflects the fact that *i* (and) is a coordinating conjunction and that it may coordinate the nouns *SNS* (Slovak National Party) and *republiku* (republic). The problem is that our system does not apply any preference for parsing compound expressions (as *i nadále*), in this case it allows to parse both parts of the compound expression as independent words, with the adverb *nadále* depending on the adjective *samostatnou* (independent). Fig. 11.29 then describes the reading which would be probably preferred by human readers.

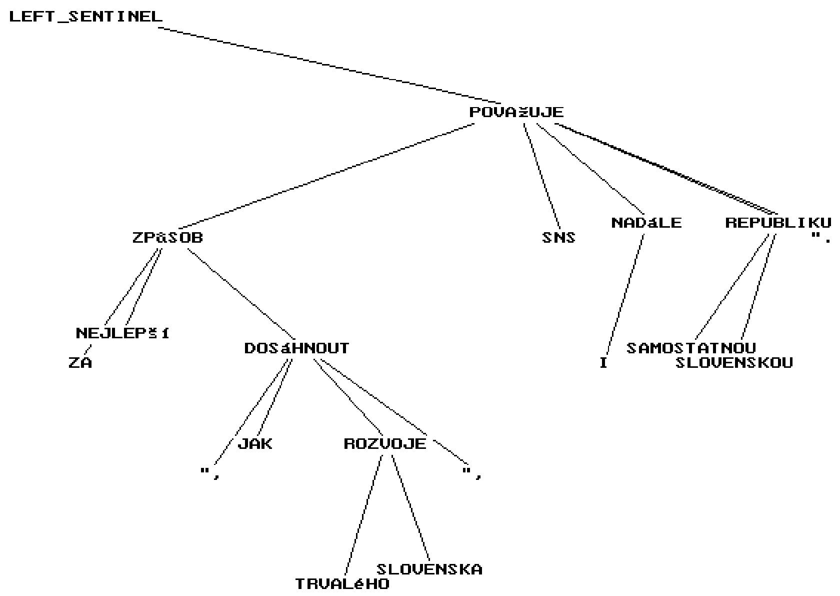

Fig. 11.29. Second tree representing the syntactically correct variant of the sentence (9)

The ill-formed variant of this sentence does not contain any commas. In the syntactic tree this fact is reflected by the system through a syntactic inconsistency between the verb *považuje* and the infinitive form of the verb *dosáhnout.* The system, of course, is not able to decide if there are missing commas or if the error is caused by an improper verbal participant as a filling of the nonexistent slot in the frame of the main verb – in both cases the syntactic structure obtained would be the same. The processing of the ill-formed variant of the sentence took 0,71s, 3 trees were created and 666 items were derived. There is one more difference between the trees representing the ill-formed variant of the sentence and the trees describing the original well-formed sentence. It is the difference in the attachment of the free modifier *Za nejlepší způsob* (As the best way). It is caused by the application of the keyword CLOSEST in the metarule describing the free modifier attachment from the left-hand side of the governing word. This keyword blocks the attachment of the free modifier to the main verb of the sentence and thus effectively reduces the number of results obtained for the ill-formed variant of this sentence.

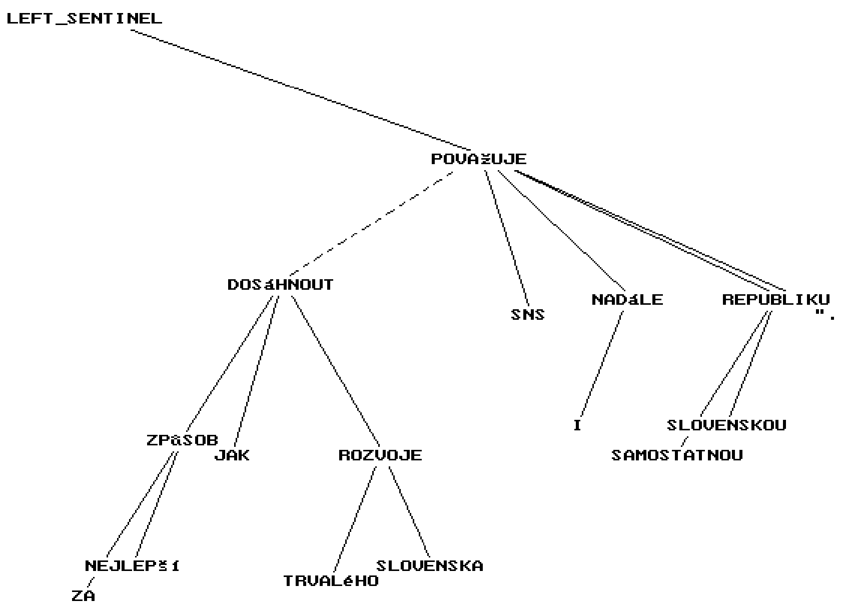

Fig. 11.30. One of the trees representing the syntactically ill-formed variant of the sentence (9)

Sentence (10)

Zasedání mohou začít být svolávána prostřednictvím generálního tajemníka.

[Meetings can begin to_be summoned through_the_mediation of_general secretary.]

(Meetings can begin to be summoned through the mediation of the general secretary.)

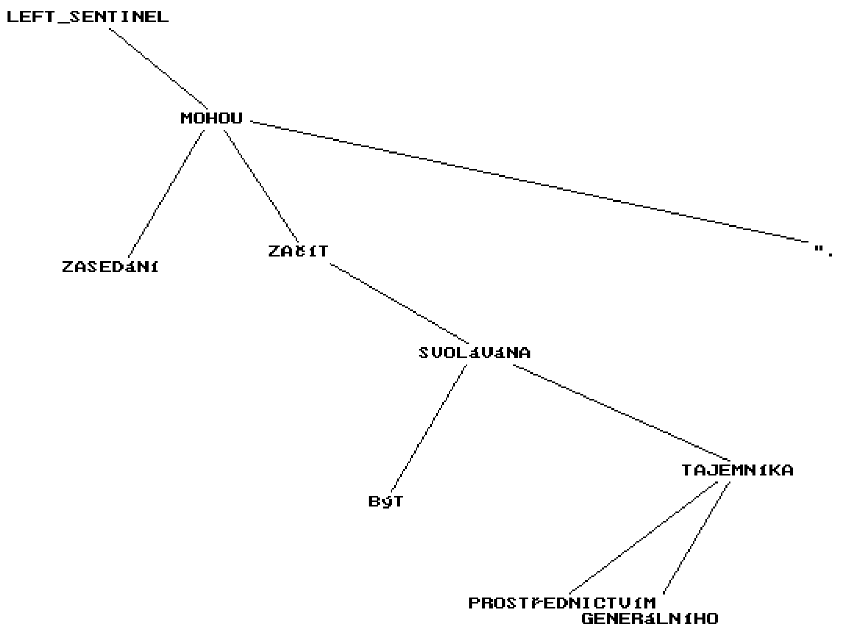

Fig. 11.31. One of the trees representing the syntactically correct variant of the sentence (10)

This sentence contains in principle two phenomena not contained in the previous sentences. The more obvious of these two is a passive construction combined with a "chain" of modal and phase denotative verbs in the infinitive form. At first sight this construction may seem to be a bit artificial, even though it is grammatical, but similar constructions sometimes appear in all kinds of texts. The main problem of this construction is that the number of embedded infinitive verbs in the "chain" is theoretically not limited (cf. *Alena se chtěla pokusit začít být vážná* (Alena wanted to try to start to be serious)) and that the agreement in number and gender between the subject and the passive verb form or adjective in verbo-nominal predicate must be preserved. The relevant metarules handling the attachment of infinite verbs to the valency frame of the governing verb take care of the propagation of the gender and number of the passive verb up through the whole chain of infinitive verbs to the node(root) where it is possible to check the subject-predicate agreement.

The second interesting phenomenon is a secondary preposition *prostřednictvím* (through the mediation of, by means of) which is in this sentence ambiguous and it is partially responsible for the number of parses (6), as demonstrated in the figures (Fig.11.31 and Fig.11.32) showing sample syntactic trees of this sentence. The problem of differentiating between noun and its conversion into secondary prepositions is not a syntactic problem. The morphological analysis must be able to recognize certain nominal forms as ambiguous and provide the relevant data allowing to treat these forms as prepositions. The second factor increasing the number of trees is the attachment of the period to all verbs in the sentence except for the auxiliary verb *být* [to be]– cf. the sentence (5). The processing took 0,11s and 153 items were created.

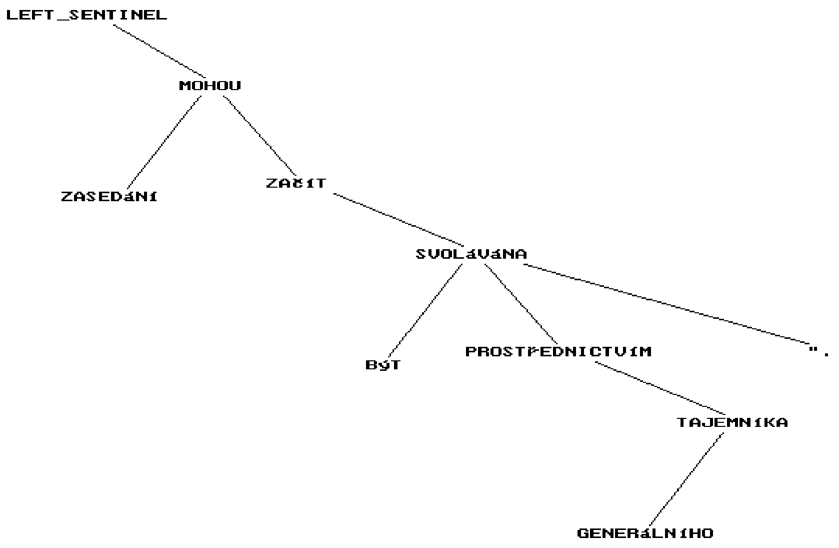

Fig. 11.32. Second tree representing the syntactically correct variant of the sentence (10)

The ill-formed variant of the sentence:

Zasedání mohou začít být **svolávány**[fem.,inan. pl.] *prostřednictvím generálního tajemníka.*

[Meetings can begin to_be summoned through_the_mediation_of general secretary.]

(Meetings can begin to be summoned through the mediation of the general secretary.)

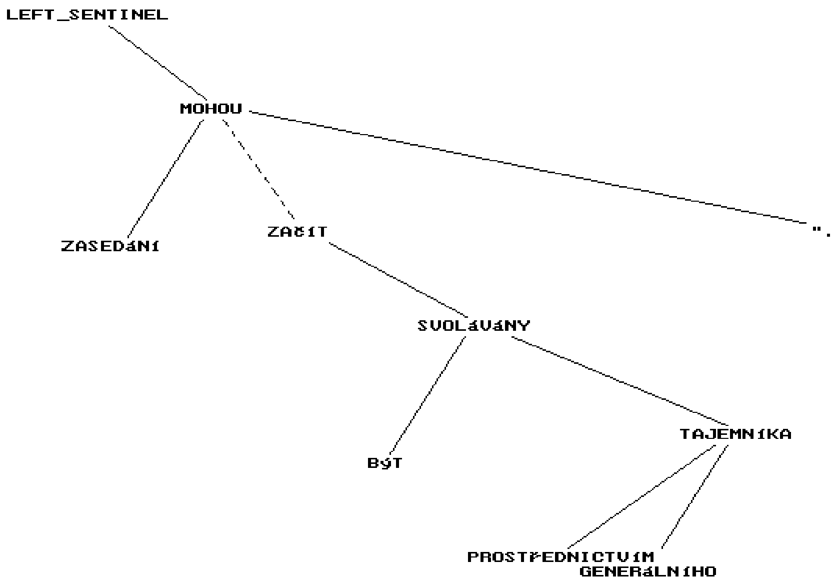

Fig. 11.33. One of the trees representing the syntactically ill-formed variant of the sentence (10)

The gender of the passive verb was changed in order to check the main phenomenon (from our point of view) contained in this sentence: the subject – predicate agreement spanning across the "chain" of infinitive verbs. As it is shown in the figures displaying the syntactic representation of the ill-formed sentence, the agreement error is manifested by syntactic inconsistencies on both sides of the root of the tree. That is a correct (acceptable) result because the system is not able, of course, to decide where the source of the error is without a subsequent evaluation of all syntactic inconsistencies in all trees.

The presented syntactic trees represent a sample of 12 trees provided by the system. The processing time of this variant of the sentence is 0,50s {0,22} and 137 items were derived.

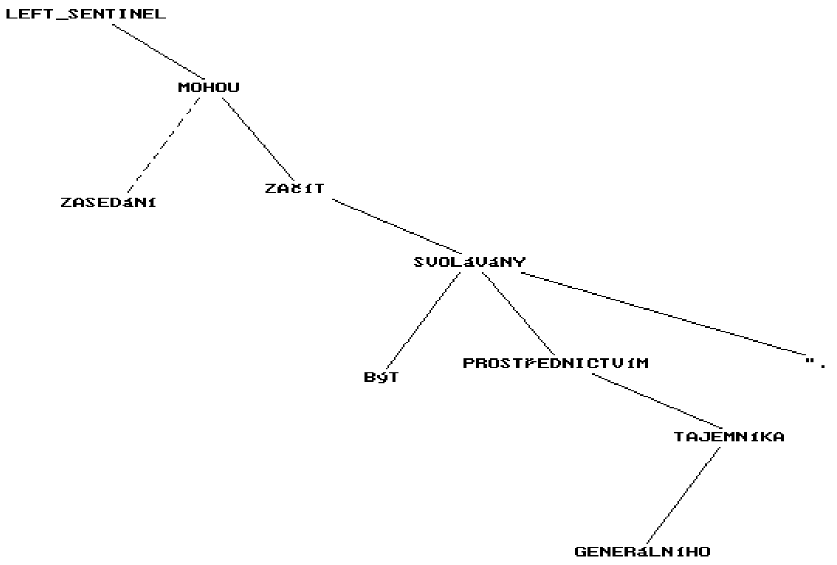

Fig. 11.34. Another tree representing the syntactically ill-formed variant of the sentence (10)

Sentence (11)

Každá vysoká smluvní strana může předložit prostřednictvím generálního tajemníka Rady Evropy komisi každé údajné porušení ustanovení této úmluvy jinou vysokou smluvní stranou.

[Each high signatory party can submit through_the_mediation_of general secretary [of] Council [of] Europe [to] commission each alleged violation [of] enactment [of] this declaration [by] other high signatory party.]

(Each high signatory party can submit to the commission through the mediation of the general secretary of the Council of Europe each alleged violation of the enactment of this declaration by any other high signatory party.)

The sentence was taken from the Czech translation of an official document of the Council of Europe. The main problem with this sentence is a high number of ambiguous words in the long chain of nouns, which results in a high number of possible parses even though the sentence seems to be quite unambiguous from the point of view of the human reader. The ambiguity concerns the words *každá* (*each* – this pronoun may also play a syntactic role of a noun) and *vysoká* (*high* – this adjective can be also used as a noun in the sense of a forest game such as a deer or, slightly colloquially, in the same sense as *university*). The situation is complicated by the fact that the ambiguous noun phrase is used twice and thus the number of parses is doubled. The parser provides 112 trees, the processing takes 6,53s and 3413 items are derived.

There is no effective solution to this problem in our metagrammar. The best way of dealing with ambiguities of this type is to insert a kind of preprocessing containing certain heuristics capable to suppress less probable readings using different kinds of information (empirical or probabilistic). This preprocessing phase would filter out spurious readings and provide the parser with the best estimation of which variants should be used. If the parser fails in the positive projective phase, then instead of invoking the second phase (positive nonprojective and negative projective) it will take into account *all* variants of the input words (including those previously discarded) and it will try to parse in a positive projective way once more. It is clear that this strategy would help only in case the sentence is syntactically well-formed, but even that would be an improvement.

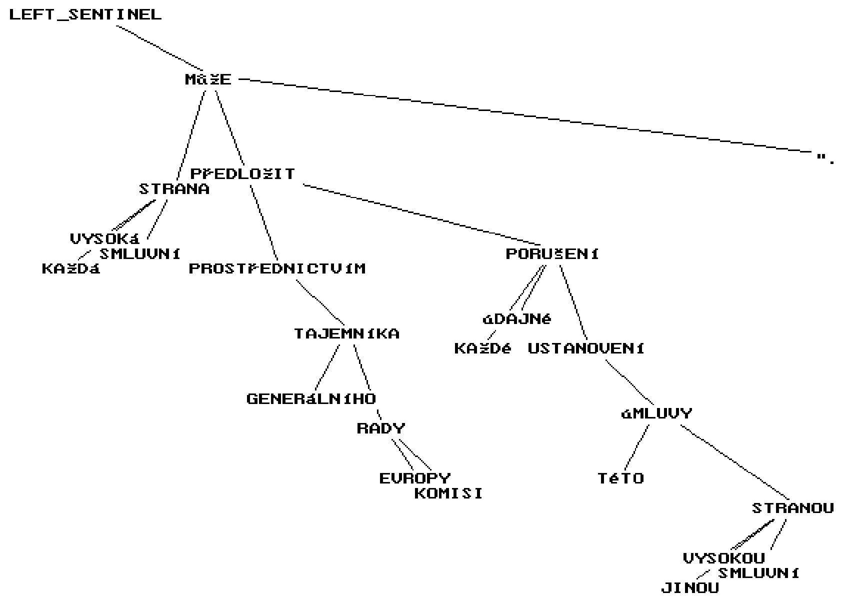

Fig. 11.35. One of the trees representing the syntactically correct variant of the sentence (10)

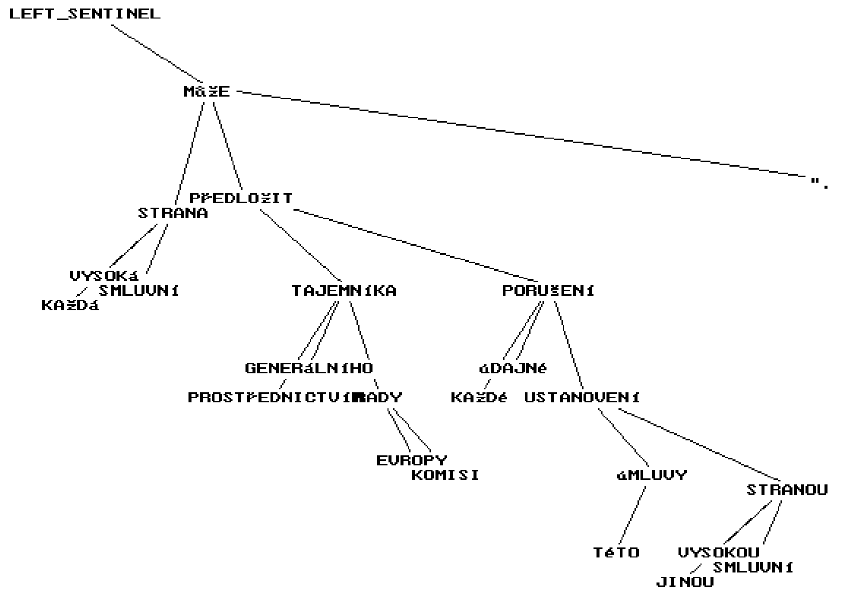

Fig. 11.36. Another tree representing the syntactically correct variant of the sentence (10)

The ill-formed variant of the sentence contains a plural form *mohou* instead of the singular *může* and thus the subject-predicate agreement is violated in a similar way as in the previous sentence:

Každá vysoká smluvní strana **mohou** *předložit prostřednictvím generálního tajemníka Rady Evropy komisi každé údajné porušení ustanovení této úmluvy jinou vysokou smluvní stranou.*

(Each high signatory party *can*[pl.] submit to the commission through the mediation of the general secretary of the Council of Europe each alleged violation of the enactment of this declaration by any other high signatory party.)

There are two reasons for checking this type of disagreement once more in this sentence. The first reason is the high frequency of agreement errors in texts written by native speakers, the second is the necessity to demonstrate how the system handles the situation when the subject is a slightly more complex nominal group. As it is shown in Fig.11.37, the system is able to find the agreement inconsistency and to reconstruct the syntactic tree of the sentence. This tree is one of 112 trees and 7076 items derived. The processing takes 28,72s.

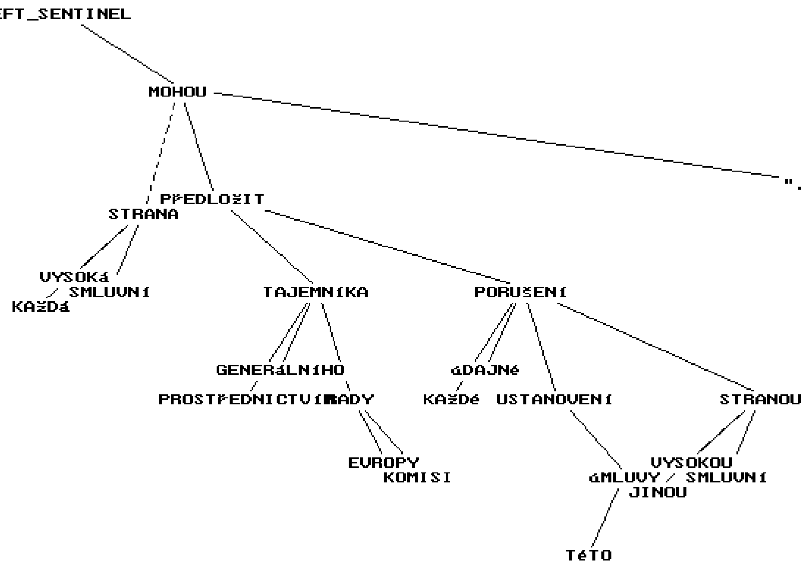

Fig. 11.37. One of the trees representing the syntactically ill-formed variant of the sentence (10)

Sentence (12)

Funkční období soudců, která trvají šest let, nelze měnit.

[Functional periods [of] judges, which last six years, it_is_impossible [to] change.]

(The functional periods in office for judges, lasting six years, are not a subject of change.)

The twelfth sentence of the testbed is the first one containing a nonprojective construction. Apart from that it also contains a noun phrase with a basic numeral and a predicative adverb *nelze* (it is not possible) which has similar syntactic properties as a verb.

One of the unpleasant effects of the mechanism dealing with nonprojective constructions is the number of parses we get even when we apply a very strong constraint on the number of holes – if we limit the nonprojective constructions only to those containing at most one hole.

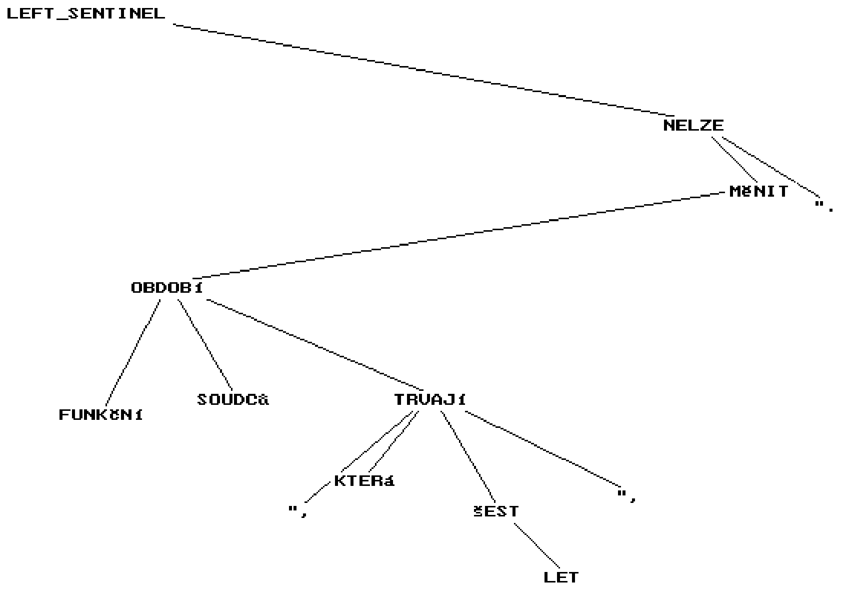

Fig. 11.38. One of the two trees representing the syntactically correct, but nonprojective variant of the sentence (11)

The problem of acceptance of noun phrases containing basic numerals is another very important problem. The acceptance of semantically incorrect phrases (*třicet sto pět tisíc stromů* (thirty hundred five thousands trees) instead of *pět tisíc sto třicet stromů* (five thousand one hundred thirty trees)) cannot be avoided at the level of surface syntax (cf. the examples: *Bylo sedm dvacet*. (The time was seven twenty.); *Stojí to pět padesát.* (It costs five fifty.); *Vyhráli jsme sedm jedna.* (We have won seven one.))

The only phenomenon worth checking at this level is the case required by the last numeral preceding the noun (in the nominative or accusative case there is a difference in cases required by numerals 1, 2, 3 and 4 and all others – nom.sg for 1, nom.pl. for 2,3,4 and gen.pl. for all others).

The predicative adverb *nelze* is handled as a verb, its specific syntactic properties are reflected in its valency frame.

The sentence has 2 parses, the processing takes 4,07s and 1240 items are derived.

The second syntactic tree differs only in the attachment of a period; it is attached to the infinite verb *měnit* (to change).

The ill-formed variant of this sentence does not contain the comma following the relative clause:

Funkční období soudců, která trvají šest let nelze měnit.

[Functional periods [of] judges, which last six years it_is_impossible [to] change.]

(The functional periods in office for judges lasting six years are not a subject of change.)

Since the predicative adverb *nelze* is treated as the verb, the parser ends up with two main verbs (*trvají* and *nelze*) not separated by any delimiter (conjunction, comma etc.). This error is quite difficult to locate and classify, even though there is a soft condition checking the presence of a comma or period following the relative clause. The syntactic inconsistencies reported by the system concern the disagreement in case between the noun and the relative pronoun and the relevant slots of the valency frame of their governing words (*období – nelze* and *která – trvají*). This is caused by the presence of the nonprojective construction in the sentence. The positive projective phase fails to create any tree for the sentence. The second phase is looking for either a projective tree containing syntactic inconsistencies or for a nonprojective tree without syntactic inconsistencies. Unfortunately, this phase succeeds and it creates four projective trees with inconsistencies. It is quite natural that these trees contain more than one inconsistency, because a nonprojective construction in a projective tree is typically a source of false inconsistencies.

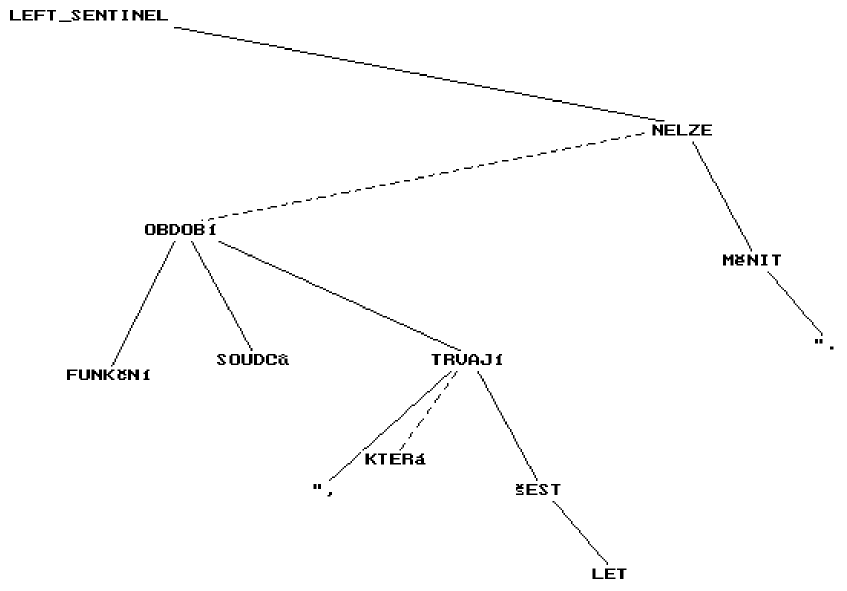

Fig. 11.39. One of the trees representing the syntactically ill-formed variant of the sentence (12)

This sentence is very interesting, because it shows that in order to get a more accurate representation of ill-formed sentences it is really necessary to include an evaluation phase, whose task would be to provide a more accurate tree based on the evaluation of all kinds of syntactic inconsistencies achieved as a result of parsing the sentence. This sentence shows very well that this task is not easy and that it will require a very extensive research in future.

The processing of the ill-formed variant takes 1,27s, 4 trees and 652 items are created.

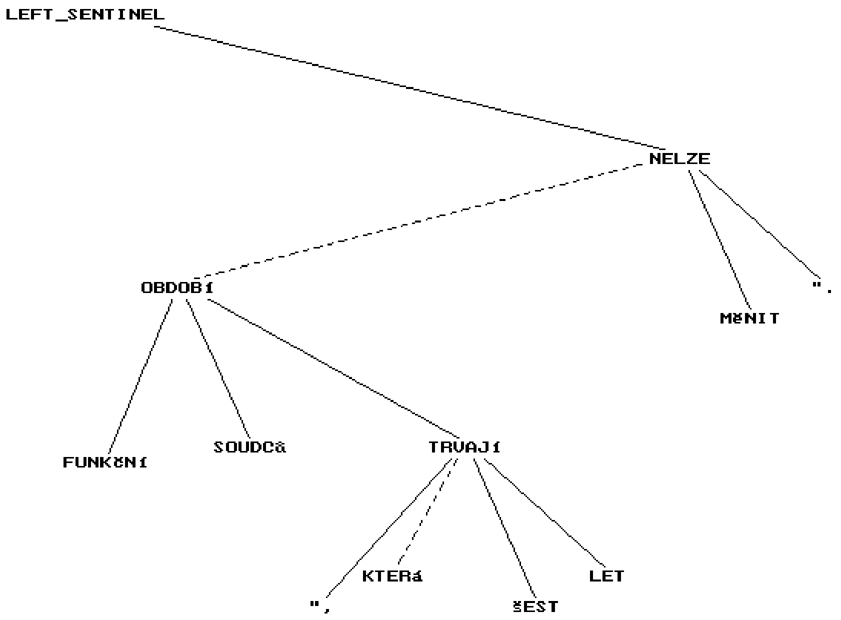

Fig. 11.40. Other tree representing the syntactically ill-formed variant of the sentence (12)

Sentence (13)

Mám pocit, že to bylo malou osvětou mezi poslanci, a to nejen v parlamentu, ale i v tisku.

[I_have feeling, that it was by_small enlightenment among deputies, and that not_only in parliament, but also in press.]

(I have a feeling that it was caused by a neglected enlightenment among MP's, not only in the parliament but even in the press.)

This sentence contains two new phenomena. The first one is the construction with *a to* (and that, namely) which modifies the preceding nominal group. The problem is that the conjunction *a* (and) is used here in the gradation sense. In order to be able to parse this construction correctly, we have to make sure that a coordinating conjunction is followed by a demonstrative pronoun and preceded by a comma. This is handled by a series of three metarules (cf. the description of the metagrammar in the previous chapter).

The second interesting construction is the pair *nejen – ale i* (not only – but also). It represents a whole class of constructions, where two words form a pair, the component parts of which are mutually related but quite distant from each other in the sentence. This class contains among others also the pairs *nejen – ale ani* (not only – nor), *buď – a nebo* (either – either), *ani – ani* (neither – nor) etc. It is very difficult, if not impossible, to parse these pairs on the basis of local context only. It would be much easier if we could check the presence of these pairs in the sentence globally and to parse the respective part of the sentence separately.

This type of constructions is another reason for our advocating the idea of a preprocessing module which would be capable to look for certain phenomena in the morphologically processed input sentence and to mark them in such a way that would make the syntactic parsing easier. In our system we have decided to solve the problem of these constructions by introducing a special attribute *pair* with the values *left* or *right.* Let us suppose that this attribute is inherited from the lemma from the syntactic dictionary of the system and that the value *left* means that a particular word may appear as the first member of some of the pairs mentioned above and the value *right* is used for a word in the position of the second member of one of the pairs.

It is evident that this solution is inferior to the solution proposed above, namely the involvement of a special preprocessing module. The module would not only be capable to check the presence of "suspicious" words in the sentence, but it also should check whether they really constitute a pair or not. The solution used in our system is a compromise between adequacy and simplicity of the metagrammar. It will parse not only the pairs really existing in the language, but also the incorrect pairs like *nejen – a nebo* [not only – either]. This solution would be unacceptable if we were concerned with checking the syntactical correctness of the sentence in the grammar checker, while for the robust parser it in fact means only a small imprecision. Let us remind that our ultimate goal is a robust parser, we would have to accept the incorrect sentence anyway, the only difference is that with our solution we are not able to point out the inconsistency or to correct it. If we liked to avoid this inadequacy, we would have to introduce a very complicated metarule (or even a set of metarules) listing all possible pairs. Such a solution would go against our initial decision that metarules should describe the interaction of groups or classes of words, not among individual words.

The processing of the sentence takes 1,10s, 2 trees and 1351 items are created.

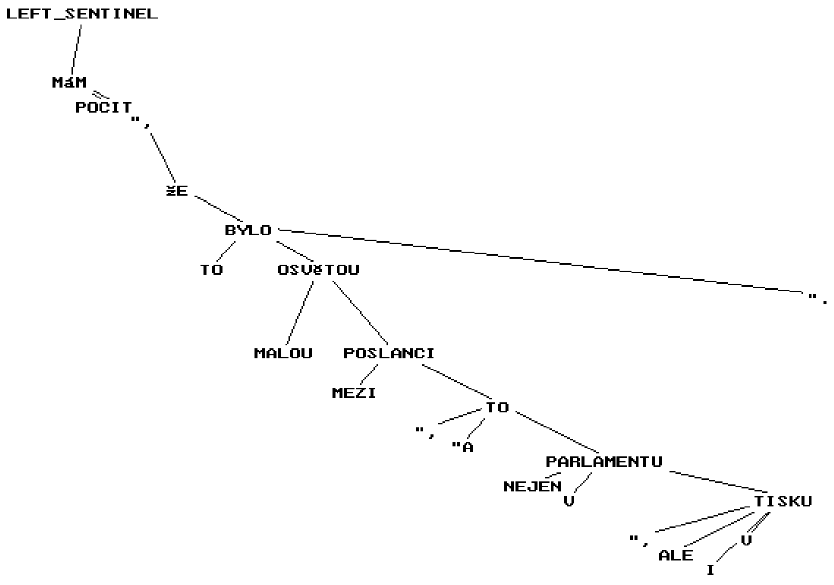

Fig. 11.41. One of the two trees representing the sentence (13)

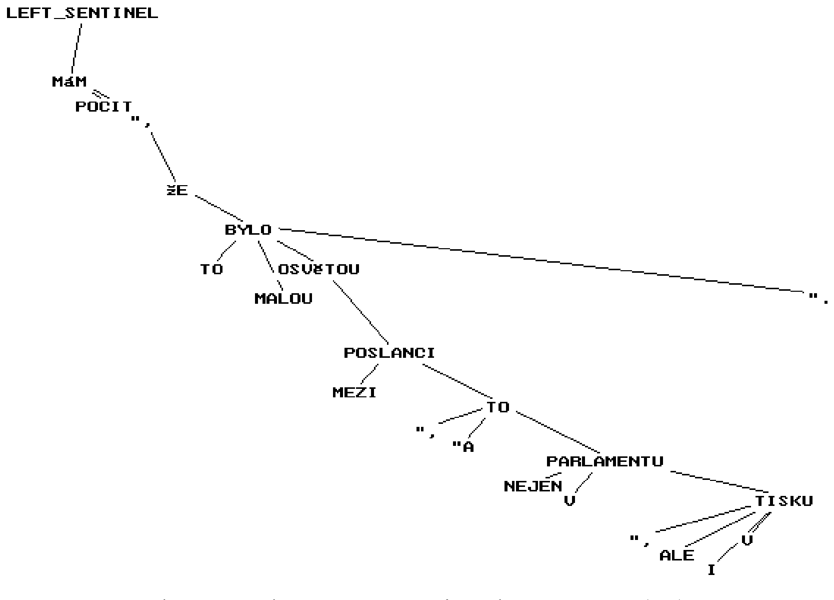

Fig. 11.42. The second tree representing the sentence (13)

The ill-formed variant of this sentence lacks the comma preceding the conjunction *a* (and). As we have already mentioned above, this kind of error is very common in Czech. The metarules handling the construction with *a to* (and that, namely) are also able to cope with this error (cf. the description of the metagrammar), as shown in the following two figures. The processing of this variant of the sentence (13) took 5,93s, 8 trees and 3298 items were created. Fig.11.43 shows the tree corresponding to that from Fig.11.41.

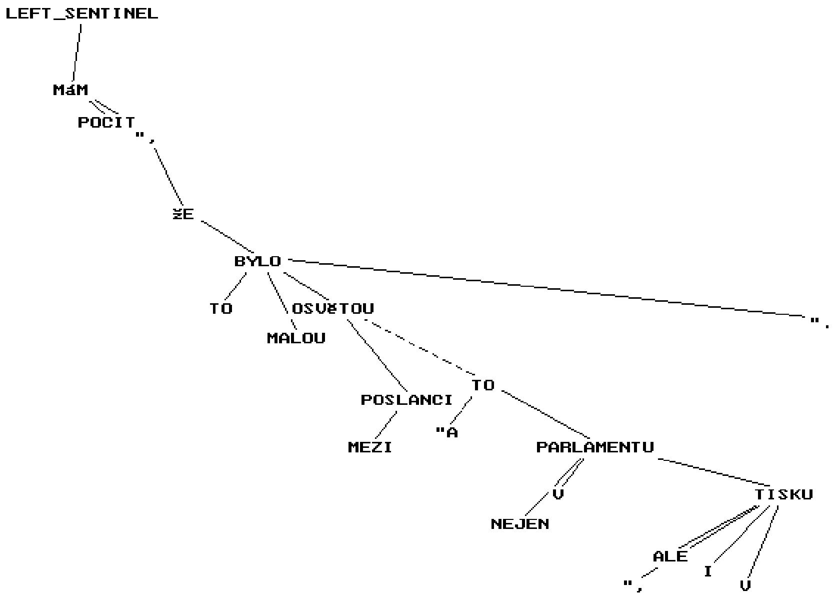

Fig. 11.43. One of the trees representing the syntactically ill-formed variant of the sentence (13)

On the other hand, Fig.11.43 contains a tree with syntactic inconsistency between the preposition *mezi* (among) and the noun *poslanci* (deputies). This was caused by a locally acceptable analysis of the coordinating conjunction *a* [and] in its proper role (coordinating the noun *poslanci* (deputies) with the pronoun *to* (it)). Due to the soft constraint requiring the case agreement of coordinated words the noun *poslanci* is considered to be in the nominative case, which conflicts with the case required by the preposition *mezi* (between). This tree represents another clue for the claim that for proper error localization and classification it is necessary to take into account the full set of results provided by negative phases of our parser due to the fact that some inconsistencies may really be very misleading.

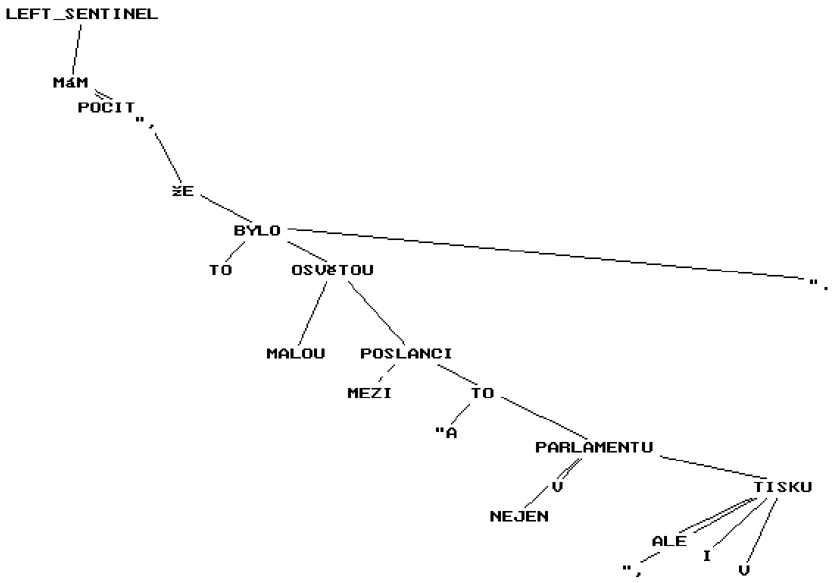

Fig. 11.44. Another tree representing the syntactically ill-formed variant of the sentence (13)

Sentence (14)

Přes to však je postup konkrétních kontraktů pomalejší, než bychom chtěli.

[In_spite_of that however is process [of] concrete contracts slower, than we_would want.]

(In spite of that, however, the process of concrete contracts is slower than we would like it to be.)

The reason why this sentence was included into the testbed is the comparison by means of the construction with an adverb *než* (than). This construction should be preceded by a comma in case it forms a clause, as is the case in our sample sentence. In other cases there should be no comma preceding *než* (than) – cf. the sentence (7).

The processing of our sample sentence took 1,98s, 900 items were derived and only a single tree was created.

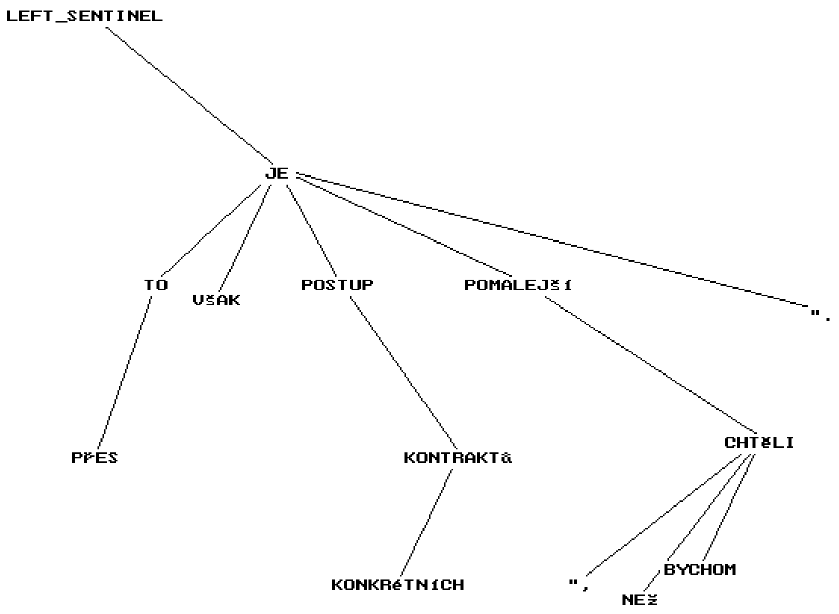

Fig. 11.45. The result of parsing the sentence (14)

In order to be able to demonstrate the difference between a single nominal group with the meaning of comparison and a subordinated clause we had to modify the sample sentence slightly more than in other cases. The modified sentence (14) with incorrectly inserted comma looks like this:

Přes to však je postup konkrétních kontraktů pomalejší, než v minulém roce.

[In_spite_of that however is process [of] concrete contracts slower, than in previous year.]

(In spite of that, however, the process of concrete contracts is slower than in the previous year.)

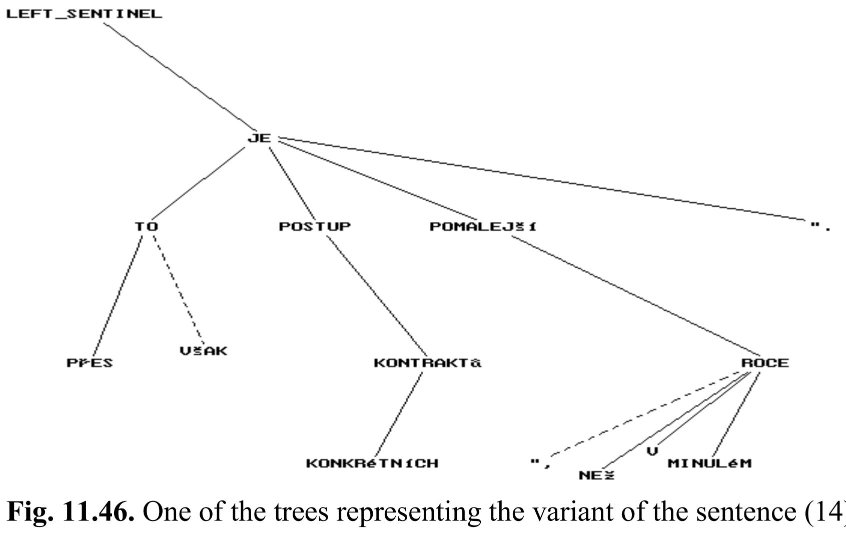

Fig. 11.46. One of the trees representing the variant of the sentence (14)

The Fig.11.46 shows a tree containing not only the expected syntactic inconsistency of a comma preceding the adverb *než* [than], but also another inconsistency. The number of syntactic inconsistencies in one tree is not limited, so the system has found a solution, which is clearly inferior to that shown on Fig.11.47. Similarly as with the previous sentence, the syntactic error should be identified by an evaluation module on the basis of both trees obtained as a result of parsing. 1043 items were derived in 2,31s during the parsing of this sentence.

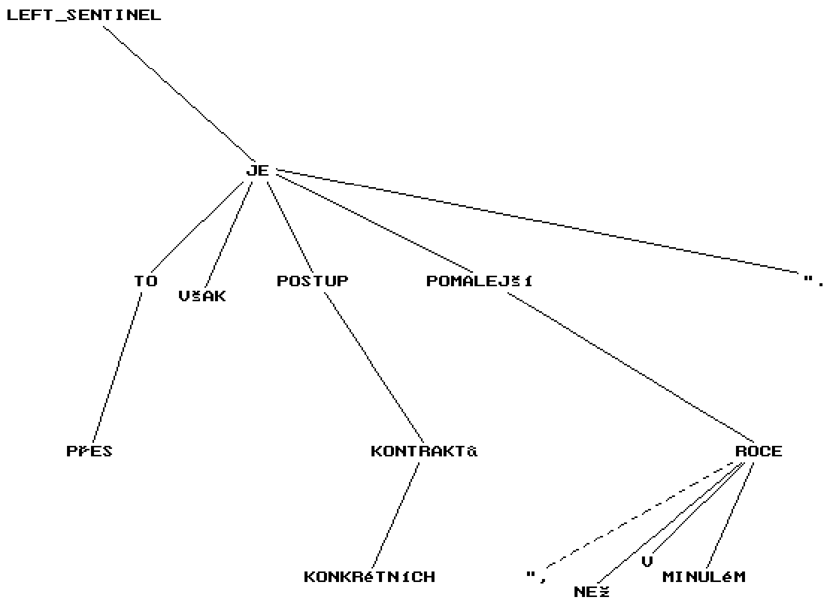

Fig. 11.47. The second tree representing the variant of the sentence (14)

Sentence (15)

Tuto knihu jsem se mu rozhodl dát k vánocům.

[This book I_am Refl. decided to_give him to Christmas.]

(This book I have decided to give him to Christmas.)

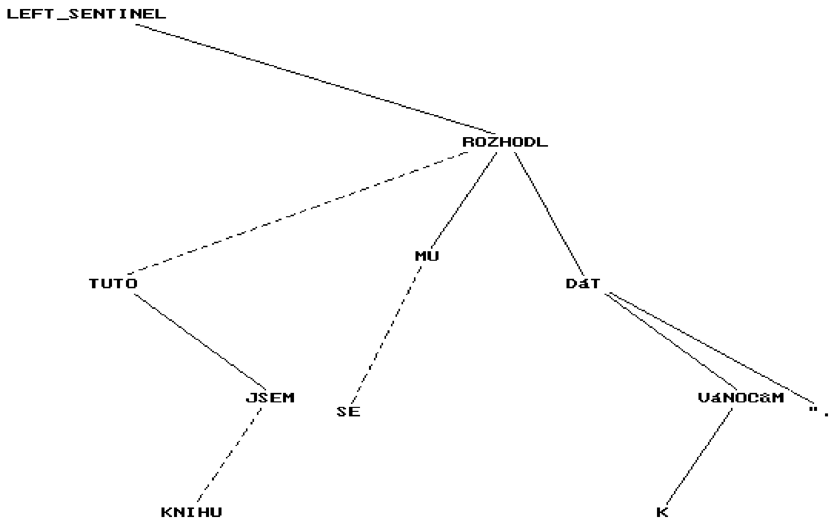

Fig. 11.48. One of the trees obtained as a result of parsing of the sentence (15)

This sentence is very interesting due to its complexity. Not only is it nonprojective, but the number of holes in it equals two and thus it exceeds the limit (one hole) set as a parameter of the interpreter in our system. It is then no wonder that the system is not able to parse the sentence with this limit without reporting syntactic inconsistencies. The system creates 11 trees. It derives 1347 items in 3,46s. The two examples of trees in Fig. 11.48 and 11.49 illustrate the fact that the inadequate limit on the nonprojectivity of the sentence leads to inadequate trees.

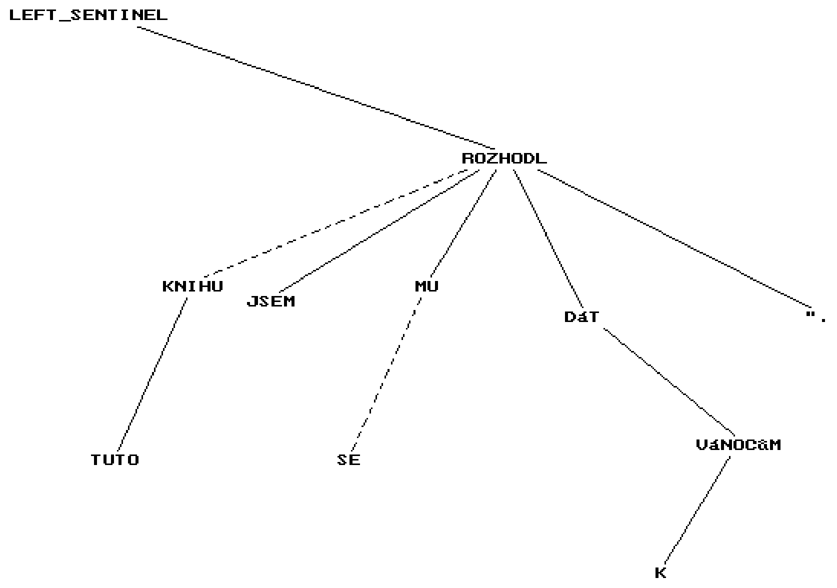

Fig. 11.49. Another tree obtained as a result of parsing of the sentence (15)

If we raise the limit of holes allowed to two, the results will change. In order to do this, we have to use a newer version of the interpreter, namely the version currently being developed by Tomáš Holan in his thesis [Holan 01]. The question might arise why we didn't use the newer version as a basic testing and development tool for our metagrammar. The answer is very simple – the process of building and testing a metagrammar for natural language parsing is very long and complicated. During the work on the LATESLAV project the process of creating and testing a metagrammar to a large extent influenced also the process of the implementation of the software environment. In LATESLAV we have at the same time tested the method, the software environment and, last but not least, also individual metarules of the metagrammar. For the robust parser it was necessary to use more stable environment and to concentrate on the development and testing of the metagrammar only – therefore we had to stick to the version of the software environment we had at our disposal when we have started the work. The development of the software environment and also the development of the theory underlying it went on a slightly independent track (cf. [Holan 01]). That is the reason why we did not mention the new version of the software environment until we had to cope with a general constraint (dNg <= 1) which was set too low in the version of the interpreter we are using. Fig. 11.50 shows the result of parsing with the new version of the software environment. The new software environment uses also different font, therefore Fig. 11.50 slightly differs from all previous figures in this chapter.

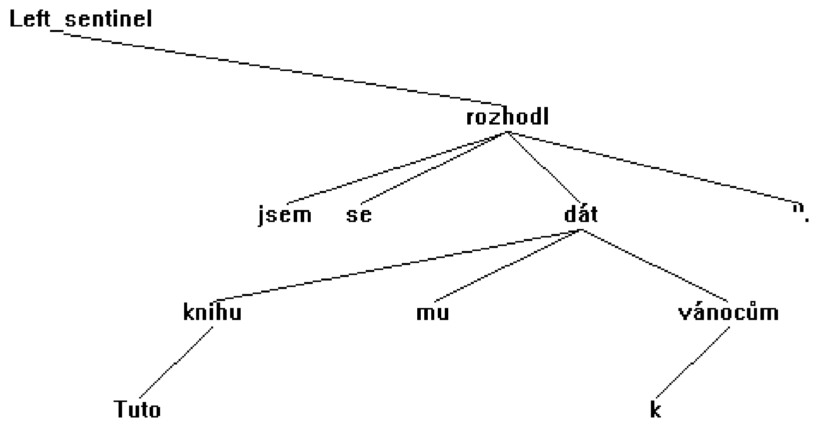

Fig. 11.50. The result of parsing with a current version of the interpreter

The fact that the interpreter we are using for our metagrammar is not able to parse the sentence (15) in an adequate manner is also a reason why we did not include an ill-formed variant of this sentence into the testbed. The total number of sentences in the testbed thus currently remains equal to 29 (fifteen syntactically well-formed sample sentences and fourteen of their ill-formed variants, as they were presented in this chapter).

Chapter 12: A Method for Analyzing Clause Complexity

In the previous chapters we have described an attempt to handle robust parsing of Czech in a consistent way (both linguistically substantiated and computationally effective) which will account for typical features of the language (a high degree of word-order freedom, a relatively high frequency of nonprojective constructions etc.). The main idea of this approach is a separation of individual steps (phases) of the parsing, where the power of each phase (except the last one) is to some degree constrained. The constraints allow for a more efficient processing and also to some extent block the creation of superfluous (and computationally expensive) results of parsing (for example, a projective reading without syntactic inconsistencies is preferred).

This general approach has proved to be useful and it seems that it is a good idea to pursue this direction even further and to continue dividing the parsing process into even more phases. This strategy may help to overcome some problems, which have been mentioned in the section describing the results of parsing individual sentences from the testbed. Those problems are to some extent coped with in the metagrammar (cf. the description of individual metarules in Chapter 9), but it is clear that the software environment used in this project does not provide sufficient means for their solution. It is also clear that they might be solved in a more efficient and elegant way. One of the weak points is the inability of our interpreter (quite intentional, allowing to write less complicated metarules which are easy to interpret and apply) to take into account a broader context inside the sentence than just two items. Even the possibility to apply a metarule to two items which are not immediately adjacent does not provide the power necessary for a fast and smooth parsing of certain sentences. In this chapter we would like to describe an outline of a method, which could bring a significant improvement to the approach described in previous chapters. The main idea is to divide the sentence into clauses before the start of the syntactic parsing. This method would substantially simplify the parsing process, in fact it will greatly reduce the complexity connected with the high degree of word-order freedom in Czech due to the fact that it would make it possible to process individual clauses more or less independently.

The endeavor to divide the parsing process into a cascade of smaller steps is not new. These strategies became quite popular recently, very interesting approaches were presented for example in [Abney 96], [Ciravegna, Lavelli 99] or in [Brants 99]. The advantage of working with a cascade of specialized parsers instead of having one very complex general parser is quite obvious – the complexity of the task is substantially reduced and the parsing process is speeded up. The major problem is how to divide such a complex task into individual steps of the cascade so that the division would not negatively affect the quality of parsing results.

12.1 Motivation

One of the serious issues of parsing is the problem of parsing embedded subordinate clauses. In the literature these two terms (embedded and subordinated) are often used as synonymous. As we have already mentioned, in this thesis we would like to use the term "embedded" in a slightly different manner than the term subordinated. For the sake of more precise explanation we have decided to distinguish between subordinate and embedded clauses. Subordinate clause is a clause the head verb of which "depends" on a word contained in another clause. The embedded clauses are those subordinate clauses which are inserted into the middle of another clause and which thus are at the surface level followed by a nonempty tail of the governing clause.

The parsing of embedded clauses has to deal with many problems specific to each type of embedded clauses (for example the problem of determining the antecedent of an embedded relative clause), but there are also some difficulties common to all kinds of embedded clauses.

The first one is the problem of determining where the clause actually ends. Without a reliable information about the end of the clause it is usually necessary to take into account all possible variants (of parsing of a local subsequence of input words), at least until the moment when the parser has at its disposal enough information, which allows to discard irrelevant variants. The problem of our approach is that the relevant information, which flows from leafs to the root of the syntactic tree (bottom up), may be available quite late and thus the parsing process is burdened by a large number of variants, some of which are definitely irrelevant for a given sentence since the moment of their creation.

The problem of determining the end of the clause is also related to the second serious problem of embedded clauses, the task of finding an initial part of the governing clause to which the part following the embedded clause belongs. In less complicated sentences containing one main and one embedded clause this question is in fact a transformation of the previous one – the section following the embedded clause is either a part of the governing clause or a part of the embedded one (in this case it would, in fact, not be an embedded clause at all, in the sense we understand the term "embedded"). If the parser is capable to determine where the embedded clause ends, it is also able to analyze the section following the embedded clause together with the governing clause. Slightly different is the situation in the case where several subordinated clauses are embedded (nested) into each other. Let us demonstrate it by means of the following scheme:

$M\ S_1\ S_2 \ldots S_n\ T_?$,

where M represents (an initial part of) the main clause, S_i ($1<=i<=n$) are (initial parts of) subordinate clauses and $T_?$ is the remaining part (tail) of the clause. For each number i ($0<=i<n$) the sentence S_{i+1} is subordinated to the clause S_i. The variable ? in the index of T stands for either the letter M (in case the tail belongs to the main clause) or an integer from the interval <1;n>.

Let us, for example, consider the sentence (12) from the testbed:

Funkční období soudců, která trvají šest let, nelze měnit.

(Functional periods of judges which last six years may not be changed.)

This sentence has the form of $M\ S_1\ T_M$, where the index $_M$ represents the fact that the remaining part (tail) of the complex sentence constitutes a part of the main clause. Let us now look more closely at the process of parsing this sentence.

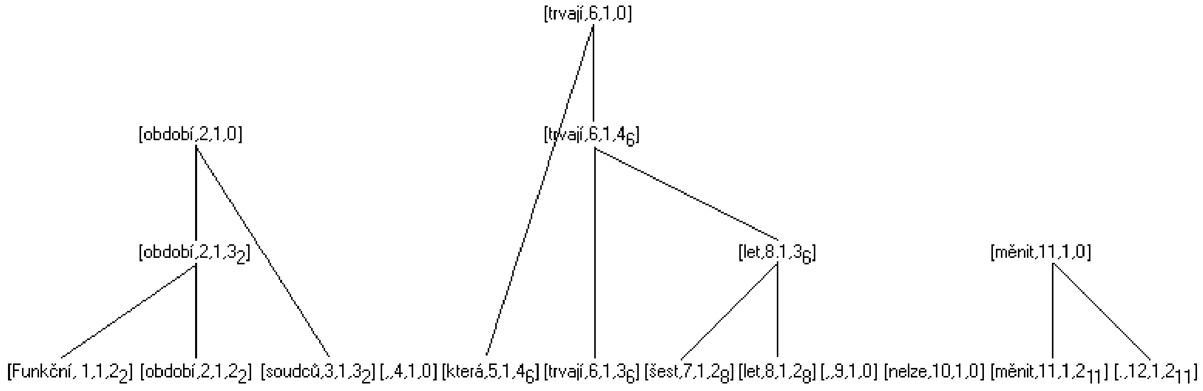
[trvají,6,1,0]
[období,2,1,0]
[trvají,6,1,4_6]
[období,2,1,3_2]
[let,8,1,3_6]
[měnit,11,1,0]
[Funkční, 1,1,2_2] [období,2,1,2_2] [soudců,3,1,3_2] [,,4,1,0] [která,5,1,4_6] [trvají,6,1,3_6] [šest,7,1,2_8] [let,8,1,2_8] [,,9,1,0] [nelze,10,1,0] [měnit,11,1,2_11] [.,12,1,2_11]

Fig. 12.1. One of the sets of DR-subtrees created during parsing

Fig. 12.1 displays one possible combination of subtrees of a DR-tree created in the process of parsing our sample sentence. The subtrees more or less cover individual segments of the sentence separated by the two commas (both commas block the analysis of longer sequences at the beginning and at the end of the sentence). This is, of course, not the only possible set of subtrees created during parsing, but this one is best suited for our purposes. Notice that the words *nelze měnit* [it_is_not_possible to_change] are not yet connected – the reason is that the initial part of the sentence in fact depends on the infinitive *měnit* [to_change] and thus (in a "correct" DR-tree) this infinitive cannot be attached by means of an oblique edge to the word *nelze* [it_is_not_possible] until the object is attached to it. The parsing may then continue as is shown on Fig 12.2:

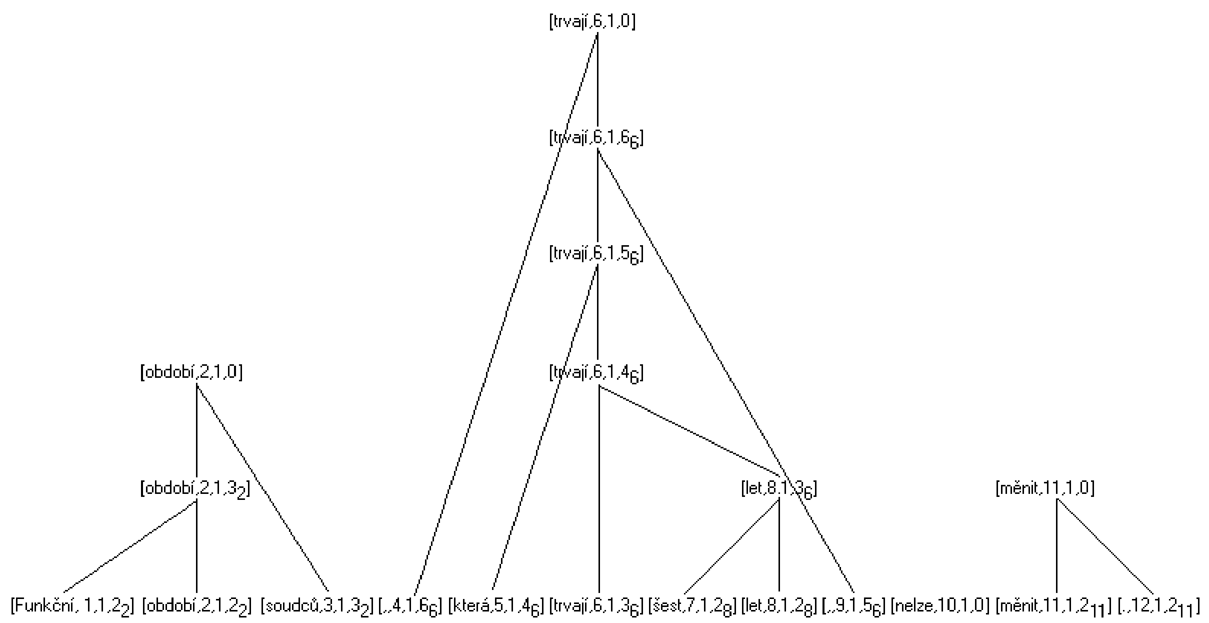
[trvají,6,1,0]
[trvají,6,1,6_6]
[trvají,6,1,5_6]
[období,2,1,0]
[trvají,6,1,4_6]
[období,2,1,3_2]
[let,8,1,3_6]
[měnit,11,1,0]
[Funkční, 1,1,2_2] [období,2,1,2_2] [soudců,3,1,3_2] [,,4,1,6_6] [která,5,1,4_6] [trvají,6,1,3_6] [šest,7,1,2_8] [let,8,1,2_8] [,,9,1,5_6] [nelze,10,1,0] [měnit,11,1,2_{11}] [.,12,1,2_{11}]

Fig. 12.2. The DR-subtrees after the attachment of both commas

Fig 12.2 displays the situation in which the whole embedded clause including the two commas (at its beginning and at its end) has already been parsed. The parsing process then continues with the attachment of the subordinated clause to the governing noun (*období* – periods). At this point the subordinate clause no longer plays any role in the parsing of the sentence. There is no difference whether there is (or is not) any subordinate clause in the sentence — the subsequent analysis of the nonprojective construction *období nelze měnit* (periods may not be changed) is independent of this fact.

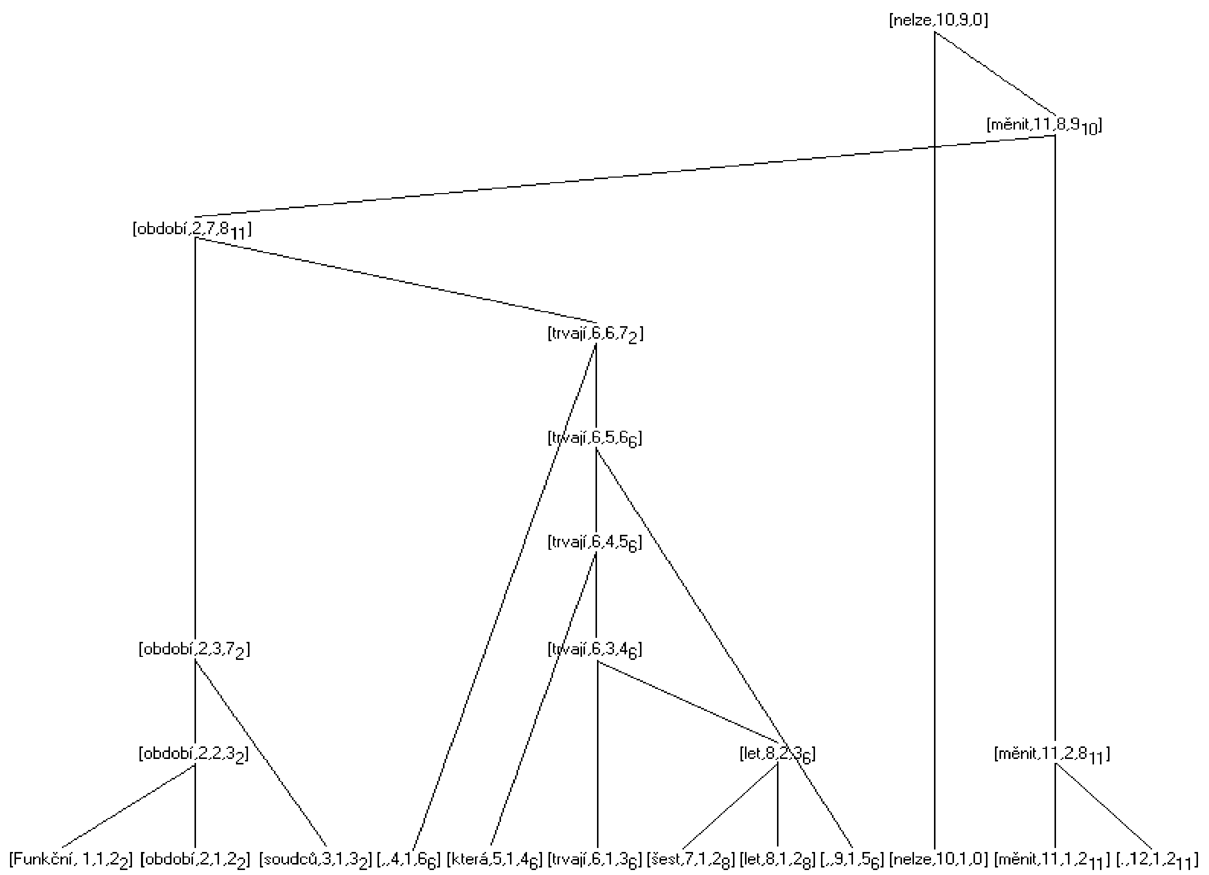
[nelze,10,9,0]
[měnit,11,8,9_{10}]
[období,2,7,8_{11}]
[trvají,6,6,7_2]
[trvají,6,5,6_6]
[trvají,6,4,5_6]
[období,2,3,7_2]
[trvají,6,3,4_6]
[období,2,2,3_2]
[let,8,2,3_6]
[měnit,11,2,8_{11}]
[Funkční, 1,1,2_2] [období,2,1,2_2] [soudců,3,1,3_2] [,,4,1,6_6] [která,5,1,4_6] [trvají,6,1,3_6] [šest,7,1,2_8] [let,8,1,2_8] [,,9,1,5_6] [nelze,10,1,0] [měnit,11,1,2_{11}] [.,12,1,2_{11}]

Fig. 12.3. One of the DR-trees representing the syntactic structure of the whole sentence

This sentence may serve as an example that uncomplicated complex sentences may be parsed successfully (and efficiently) even without using information about the broader (sentential) context.

Let us now slightly modify this sentence (and make it more complex):

Funkční období soudců, která trvají šest let, z nichž jeden rok soudce vykonává funkci předsedy soudu, nelze měnit.

(The functional periods of judges, which last six years, one of which each judge serves in the function of the court president, may not be changed.)

This sentence has the form of M S_1 S_2 T_M . It is more difficult to analyze than the previous one. The local analysis of segments separated by commas is similar as in the previous case, also the analysis of the embedded clauses is quite smooth and uncomplicated. The difficulties begin when the parser creates the DR-subtree for the relative clause S_1. This in fact corresponds to the situation from Fig 12.2 The DR-subtree of the clause S_1 may now be parsed as a dependent item of its antecedent noun (*období* – periods) and thus it may no longer be used in the parsing process. In this case, from now on everything what the parser will combine with this DR-subtree represents an invalid structure. A similar problem will appear later on when the parser creates DR-subtrees representing the clause S_2. This clause will also be attached to its antecedent and the parser will try to combine the relative clause S_1 with the remaining part of the main clause (T_M). Thus it will create several more invalid items, making the parsing process unnecessarily more complicated and increasing the risk of a parsing failure (due to the potential lack of available resources).

In the metagrammar described in Chapter 9 we have introduced one possible (partial) solution of, at least, the problem of determining the end of the subordinate clause. We are using a special kind of auxiliary "technical" attributes storing the necessary information about a (possible) end of a clause for later use (cf. the attribute EOS etc.). Although the sample sentences show that such an approach may solve certain problems, it is definitely not the most elegant and effective solution. There are at least two weak points, which motivate us to try to find an alternative way.

The first one is connected with the fact that from the point of view of a narrow local context it is often impossible to rule out the creation of subtrees representing “dead ends” of parsing a particular sentence. It is necessary to keep (and to process further) all subtrees representing all possible variants of parsing a particular piece of the sentence, even though it is clear that some of them must be superfluous.

These subtrees are being tested for the application of grammar rules and, even worse, in case all subtrees representing a particular section of the sentence are acceptable in the narrow local context, they are combined with other subtrees until it is possible to decide which of the candidates is appropriate in the given broader context. In the discussion of our sample sentence above we have already mentioned a disastrous effect on parsing efficiency, which sometimes even results in a failure of the parser due to the lack of free resources available.

The second important problem caused by the insertion of special auxiliary technical attributes is the necessity to propagate these attributes bottom up through the DR-trees. This propagation is handled automatically on vertical edges of DR-trees due to the fact that the governing node inherits all attributes and their values along the vertical edges by default.

More complicated is the case of transferring the important information along oblique edges of DR-trees. This transfer must be explicitly stated in the grammar for each separate attribute whose value should be transferred. If the grammar is being built incrementally, it is very often the case that adding a new rule to the already existing grammar means (for the author of the grammar) to go through all existing rules and modify them accordingly. This process of thorough checking of an existing grammar becomes very often a major source of inconsistencies and errors that then have a negative effect on the quality of parsing. The more auxiliary attributes and metarules the metagrammar contains, the higher is the risk associated with any new metarule added to the grammar. The sections handling the problem of transfer of values of technical attributes may be found in our grammar for example in the third, sixth, elevenths, thirteenth, sixteenth, eighteenth etc. metarules. In all these metarules it is the value of the attributes *eos*, *eoc* and *end* (indicating that the end of a clause or sentence has already been processed) which is being transferred along the oblique edge of the DR-tree.

The approach we are going to propose in this chapter is structurally different in a sense that it tries to separate the important information in an earlier stage of parsing so that the metagrammar is not bothered with its transfer at all, it only uses it whenever necessary. This way is of course more natural and also more effective than the solution mentioned above, which is implemented in the current version of the metagrammar.

The main reason why our framework is not able to cope in a natural and transparent way with the problem of identification of the end of an embedded clause (or with other problems unsolvable on the basis of the local context) is the strict constraint on the general format of each metarule (AB->X, cf. Chapter 5) which makes it possible to apply metarules in a very restricted local context.

12.2 Motivating example

One possible alternative to the solution of the above-mentioned problems may extend the original idea of our robust parser (namely the idea of introducing a kind of a cascaded parser) also to some kind of preprocessing of the input sentence before the start of the main parsing phases. The preprocessing would consist of one or more phases which would process data achieved as a result of morphological analysis (in the form of lemmas accompanied by sets of morphological tags) [see Hajič 94] and lexico-syntactic data contained in the syntactic dictionary of the system.

The basic idea behind these initial phases is an assumption that every morphologically tagged sentence already contains a lot of more or less reliable information that may be directly used for the benefit of more effective and precise syntactic parsing. In some cases it is even possible to identify a grammatical error only on the basis of morphemic data supported by a set of metarules expressing certain syntactic rules which do not require a real "deep" syntactic parsing. As an example of such a construction in Czech we may use the syntactic rule saying that it is not allowed to use two verbs in finite form inside a single clause – they must be separated either by a punctuation mark or by a conjunction. If they are not separated, it is possible to issue an error message.

The most important information we are looking for in the initial phases is the information about the structure of a complex input sentence in the sense of the mutual relationships between individual clauses, the span of embedded clauses etc. We may use the fact that in Czech there are quite strict rules how clauses are separated (by means of conjunctions, punctuation marks etc.). Let us call this type of structural information a *clause structure* of the (complex) sentence. At the beginning it is important to stress that we suppose neither that the preparsing will be able to provide an unambiguous clause structure for every sentence nor that an unambiguous clause structure exists for every sentence. The aim is to create as precise an approximation of the clause structure as possible. Let us illustrate the basic idea of our method on a complex Czech sentence taken from the newspaper Mladá Fronta Dnes:

Example 12.1.

Za předpokladu, že se Šavrda, úspěšný kapitán, psycholog a kouč v jedné osobě, dočká prodloužení smlouvy, jak doufá nejen on, měl by šanci co nejdříve dostat Novák, považovaný za nejslibnější domácí talent.

[Under assumption, that <Refl.> Šavrda, successful captain, psychologist and couch in one person, will_live_to_see prolongation of_contract, as hopes not_only he, should <Cond.> chance[accus] as earliest to_get Novák, considered as most_promising domestic talent.]

(Under an assumption that Šavrda, a successful captain, psychologist and couch in one, will live to see a prolongation of the contract, as not only he hopes, should Novák, considered to be the most promising domestic talent, get his chance as soon as possible.)

The morphological analysis of Czech uses a positional system of tags, each tag consists of 15 positions, some of which may be empty. The analysis assigns each word form all relevant lemmas accompanied by a full set of morphological tags, no disambiguation is involved at this stage. The first position of the tag always represents a part of speech of a given lemma (R=preposition, N=noun, J=conjunction, A=adjective, V=verb, Z=punctuation, C=numeral etc.), the second position identifies a subgroup of a given POS (e.g. J is a subordinating conjunction while J^ is a coordinating conjunction).

The complete description of the system of tags for Czech can be found for example in [Hajič, Hladká 98]. The words of our sample sentence have got the following tags in the morphological analysis.

Za: za RR--2---------- RR--4---------- RR--7----------
předpokladu: předpoklad NNIS2-----A---- NNIS3-----A---- NNIS6-----A----
,: , Z:-------------
že: že J,-------------
se: s RV--7----------
se P7-X4----------
Šavrda: Šavrda NNMS1-----A----
,: , Z:-------------
úspěšný: úspěšný AAFP1----1A---6 AAFP4----1A---6 AAFP5----1A---6
AAFS2----1A---6 AAFS3----1A---6 AAFS6----1A---6
AAIP1----1A---6 AAIP4----1A---6 AAIP5----1A---6
AAIS1----1A---- AAIS4----1A---- AAIS5----1A----
AAMP1----1A---6 AAMP4----1A---6 AAMP5----1A---6
AAMS1----1A---- AAMS5----1A---- AANP1----1A---6
AANP4----1A---6 AANP5----1A---6 AANS1----1A---6
AANS4----1A---6 AANS5----1A---6
trenér: trenér NNMS1-----A----
,: , Z:-------------
psycholog: psycholog NNMS1-----A----
a: a J^-------------
kouč: kouč NNMS1-----A---- NNIS1-----A---- NNIS4-----A----
v: v RR--4---------- RR--6----------
v NNNXX-----A----
jedné: jeden ClFS2---------- ClFS3---------- ClFS6----------
osobě: osoba NNFS3-----A---- NNFS6-----A----
,: , Z:-------------
dočká: dočkat VB-S---3P-AA---
prodloužení: prodloužení NNNP1-----A---- NNNP2-----A---- NNNP4-----A----
NNNP5-----A---- NNNS1-----A---- NNNS2-----A---- NNNS3-----A----
NNNS4-----A---- NNNS5-----A---- NNNS6-----A----
prodloužený AAMP1----1A---- AAMP5----1A----
smlouvy: smlouva NNFP1-----A---- NNFP4-----A---- NNFP5-----A----
NNFS2-----A----
,: , Z:-------------
jak: jak NNMS1-----A----
jak J,-------------
jak Db-------------
doufá: doufat VB-S---3P-AA---
nejen: nejen TT-------------
on: on PPYS1--3-------
,: , Z:-------------
měl: mít VpYS---XR-AA---
by: být Vc-X---3-------
šanci: šance NNFS3-----A---- NNFS4-----A---- NNFS6-----A----
co: co PQ--1---------- PQ--4----------
co TT-------------
co J,-------------
co Db-------------
nejdříve: brzy Dg-------3A----
dostat: dostat Vf--------A----
Novák: Novák NNMS1-----A----
,: , Z:-------------
považovaný:považovaný AAFP1----1A---6 AAFP4----1A---6 AAFP5----1A---6

```
        AAFS2----1A---6     AAFS3----1A---6     AAFS6----1A---6 AAIP1----1A---6
        AAIP4----1A---6     AAIP5----1A---6     AAIS1----1A---- AAIS4----1A----
        AAIS5----1A----     AAMP1----1A---6     AAMP4----1A---6 AAMP5----1A---6
        AAMS1----1A----     AAMS5----1A----     AANP1----1A---6 AANP4----1A---6
        AANP5----1A---6     AANS1----1A---6     AANS4----1A---6 AANS5----1A---6
za:     za RR--2----------  RR--4----------  RR--7----------
nejslibnější: slibný   AAFP1----3A----     AAFP4----3A----     AAFP5----3A----
        AAFS1----3A----     AAFS2----3A----     AAFS3----3A---- AAFS4----3A----
        AAFS5----3A----     AAFS6----3A----     AAFS7----3A---- AAIP1----3A----
        AAIP4----3A----     AAIP5----3A----     AAIS1----3A---- AAIS4----3A----
        AAIS5----3A----     AAMP1----3A----     AAMP4----3A---- AAMP5----3A----
        AAMS1----3A----     AAMS5----3A----     AANP1----3A---- AANP4----3A----
        AANP5----3A----     AANS1----3A----     AANS4----3A---- AANS5----3A----
domácí:domácí AAFP1----1A----     AAFP4----1A----     AAFP5----1A---- AAFS1----1A----
        AAFS2----1A----     AAFS3----1A----     AAFS4----1A---- AAFS5----1A----
        AAFS6----1A----     AAFS7----1A----     AAIP1----1A---- AAIP4----1A----
        AAIP5----1A----     AAIS1----1A----     AAIS4----1A---- AAIS5----1A----
        AAMP1----1A----     AAMP4----1A----     AAMP5----1A---- AAMS1----1A----
        AAMS5----1A----     AANP1----1A----     AANP4----1A---- AANP5----1A----
        AANS1----1A----     AANS4----1A----     AANS5----1A----
talent: talent  NNIS1-----A----     NNIS4-----A----
talent  NNMS1-----A----
.:      . Z:------------
```

The morphemic information is combined with the lexico-syntactic information contained in the syntactic dictionary of the system. The syntactic dictionary provides for example the information that the verb *dočkat* should be combined with a reflexive pronoun *se.* It also provides data about the valency frames of all relevant words.

Let us now look at the sentence more closely.

1. *Za předpokladu* {The sentence does not start with a subordinate conjunction, therefore the first part of the sentence (till the first comma) belongs to the main clause. It might of course be the case that the main clause is not complete (it does not contain a verb), but that is not apparent here, it is necessary to wait until a broader context is analyzed.}
2. *, že se Šavrda* {Beginning of the subordinate clause. The fact that the morphological analysis did not recognize the proper name should not be a problem, if we take into account that an isolated unrecognized word beginning with a capital letter in Czech almost certainly represents a name. With this assumption it is possible to solve the ambiguity of *se* – in this context it is a reflexive particle, rather than a preposition.}
3. *, úspěšný kapitán* {This comma may not end the subordinate clause, for between the subordinate conjunction and this comma there is no finite verb. It is either a comma separating individual nominal groups in coordination, or a comma starting an inserted clause or a noun phrase in apposition to the preceding word.}
4. *, psycholog* {Same situation as in the previous case.}
5. *a kouč v jedné osobě* {The coordinate conjunction *a* [and] does not coordinate here whole sentences – due to the absence of a finite verb to the left of this conjunction it is only possible that it coordinates individual noun groups.}
6. *, dočká prodloužení smlouvy* {This is the final comma of the inserted coordination of attributes in apposition. The information from the syntactic dictionary also indicates that the verb *dočkat* is the most likely candidate to be combined with the reflexive particle in 2. That also means that sections 6. and 2. are on the same level, lower than section 1.}
7. *, jak doufá nejen on* {The comma followed by the subordinate conjunction *jak* indicates clearly that this clause (in fact at this moment we can't be sure that this section of the input sentence is a

complete clause, this will be clarified after the rest of the sentence is processed) is a modifier of one of the previous clauses.}

8. *, měl by šanci co nejdříve dostat Novák* {This clause does not contain any conjunction indicating its status as a subordinate clause. If we look towards the end of the input sentence, we will notice that this clause is also the last possible candidate (due to the presence of the last finite verb of the sentence) likely to be the (second part of the) main clause of the whole complex sentence. That also means that section 1, which is on the top level too, is an integral part of this clause and sections 2-7 are embedded, modifying the noun *předpokladu.*}
9. *, považovaný za nejslibnější domácí talent.* {The agreement in gender, number and case indicates that this section is a post-modifier (in apposition) of the name *Novák.*}

The comments to individual sections of the complex input sentence give us a clue to the problem of what kind of information may support our hypothesis about the mutual relationship of sections in the sentence. In this respect the most reliable information is that about the number and position of finite verbs, conjunctions and punctuation marks in individual sections. Let us now sum up the available information:

a) Sections 1 and 8 constitute the main clause of the sentence
b) Section 9 directly modifies the proper name from Section 8
c) Sections 2 and 6 constitute one subordinated clause
d) Section 2 directly modifies Section 1
e) Section 7 directly modifies Section 6
f) Sections 3, 4 and 5 are located on the same level, modifying (as appositions *Šavrda* in Section 2
g) The comma preceding Section 6 indicates that Sections 2,3,4 and 5 need not be on the same level (in the same clause) as Section 6

This knowledge, acquired on the basis of simple considerations presented above, leads to the following schemata, which capture the structural relationships of individual sections of the sample sentence by means of horizontal edges (expressing the fact that the nodes are either coordinated or belonging to a single clause) and oblique edges (expressing the relation of subordination). The numbers in circles correspond to the numbers of sections discussed above:

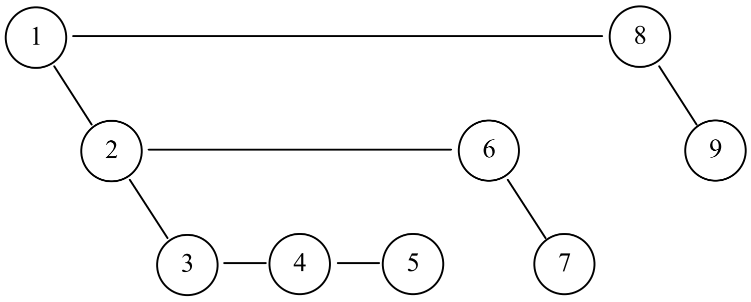

Fig. 12.4. This variant represents a coordination of sections 3, 4 and 5.

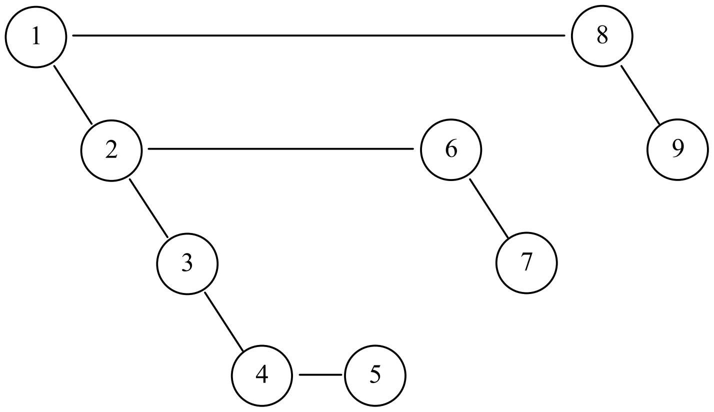

Fig. 12.5. This is also a possible variant of the structure, where only sections 4 and 5 are coordinated.

The structure of sections in the sentence is very important from the point of view of parsing adequacy and effectiveness. It allows to formulate additional constraints (to the constraints already used in all phases of the parsing) preventing a derivation of superfluous items. In certain cases these constraints may for example block an application of a metarule to items from different levels of the structure of a particular sentence etc.

The example presented above might suggest a false impression that the clause structure contains only horizontal edges combined with top-down and left-to-right oriented edges (representing the relation between the subordinate section following the governing one). The edges oriented left-to-right and bottom up (the subordinate section preceding the governing one) are also quite common in sentences where a subordinate clause precedes the governing one, as in the following example:

Example 12.2.

Aby dorazil k cíli a vyřešil po dlouhém bloudění Fermatovu větu, potřeboval Andrew Wiles jen pero, papír a obyčejnou logiku.

[In_order_to [he]_arrived towards goal and [he]_solved after long wandering Fermat's theorem, needed Andrew Wiles only pen, paper and ordinary logic.]

(In order to reach the goal and to solve the theorem of Fermat, Andrew Wiles needed only a pen, a paper and an ordinary logic.)

The sentence may be divided into following sections:

1. *Aby dorazil k cíli* {The subordinate conjunction (aby) at the beginning of the section indicates that this section does not belong to the main clause of the sentence. It also contains a finite verb, therefore this section might be a complete (subordinate) clause.}
2. *a vyřešil po dlouhém bloudění Fermatovu větu* {The coordinate conjunction and the presence of a finite verb suggest that this is also a subordinate clause, coordinated with the clause from Section 1.}
3. *, potřeboval Andrew Wiles jen pero* {This section contains the last finite verb of the whole sentence, therefore it must be the main clause (there are only three finite verbs in the sentence and the first two of them are coordinated on the "subordinate level"). The section does not contain any hint that it is not a (part of the) main clause.}
4. *, papír* {A comma followed by a noun does not itself provide a sufficient information. In this particular case it is possible to use the additional information that no finite verb follows this section. If it did and if there were a subordinate conjunction between the verb and this noun, it would be necessary to take into account the possibility that the whole complex sentence is in fact only a large nominal group governed by the noun *papír*. Without the verb following the noun it is possible to place this section at the same (or even higher) level as the previous one.}
5. *a obyčejnou logiku.* {The coordinating conjunction indicates that this section belongs to the same level as the previous one.}

These facts lead to a single possible graph:

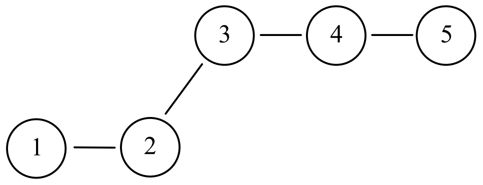

Fig. 12.6. The graph of sections of the sentence from Example 2.

Let us now define the structure of sections in a complex sentence in a more precise way. We would also like to show the connection between the structure of sections and *D-trees*.

Let us first define some important notions used throughout this section:

The term *safe verb* (*SV*) describes a word form that in the process of morphological analysis gets only tags containing the symbol for a verb (these tags begin with the letter V in the system of morphological tags used for Czech).

The term *unsafe verb* (*UV*) describes a word form which in the process of morphological analysis gets at least one tag containing the symbol for a verb and at least one tag containing a symbol for some other part of speech.

The term *delimiter list* describes a list of words, punctuation marks and other symbols that may connect individual clauses in a complex sentence in a coordinate or subordinate manner.

The term *delimiter* describes a member of a *delimiter list.*

Examples

safe verbs: byl ([he] was), *jda* (going), *píše* (writes/writing) etc.

unsafe verbs: žena {Noun(woman)/Verb(chasing)}, *tři* {Numeral(three)/Verb(rub[imperative])}, *pila* {Noun(a saw)/Verb([she] drank)} etc.

delimiters: a (and), *který* (which), *že* (that), comma etc.

Remark

It is often the case that two clauses are separated by more than one delimiter (e.g. comma followed by *že* (that)). In such a case it would be more convenient to consider the whole sequence of delimiters as a single item – let us call it a *delimiter sequence*. The introduction of this more general notion is justified also by other reasons. One of them is the fact that the sequence of words and/or punctuation marks separating two clauses sometimes does not consist only of delimiters (especially when delimiters separate the governing clause from a subordinate one). In case a delimiter is a word that may be inflected (e.g. a relative pronoun) and this word is in prepositional case, the preposition should also belong to the delimiter sequence. Another example justifying the necessity to use the broader notion of a delimiter sequence is an emphasizing adverb standing between a comma and a subordinate conjunction (cf. *, právě že* (just that)). In this context it is natural to handle the adverb as a member of the delimiter sequence.

The class of elements of the delimiter sequence which themselves are not delimiters is severely restricted in Czech and it is closely bound to the lexical value of the rightmost delimiter. For this reason it seems that the best solution would be to work with a list of word categories or individual words which may precede a certain delimiter. This list would be attached to a particular delimiter in the delimiter list.

12.3 Definitions

In the sequel an input sentence is understood to be a sequence of lexical items (word forms) w_1w_2.... w_n. Each item w_i ($1<=i<=n$) represents either a certain lexical form of a given natural language, or a punctuation mark, quotation mark, parenthesis, dash, colon, semicolon or any other special symbol which may appear in the written form of a sentence of the natural language under consideration. All items are disjunctively divided into two groups – ordinary words and members of delimiter sequences.

Definition 12.1 (Segmentation of a sentence)

Let $S = w_1w_2 \ldots w_n$ be a sentence of a natural language. A *segmentation* of a sentence S is a sequence of *sections* $D_0W_1D_1 \ldots W_kD_k$ where *section* W_i ($1<=i<=k$) represents a sequence of lexical items $w_jw_{j+1} \ldots w_{j+m}$ not containing any delimiter and *section* D_i ($0<=i<=k$) represents a *delimiter sequence* composed of items $w_Lw_{L+1} \ldots w_{L+p}$. The section D_0 may be empty, all other sections D_i ($0<=i<=k$) are non-empty. Each item w_i for $1<=i<=n$ belongs to exactly one section D_j if it is a member of a delimiter sequence or else W_j in the opposite case.

Remarks

D_k represents the final punctuation mark at the end of a sentence.

The section D_0 is usually empty in case the sentence starts with a main clause. If a complex sentence starts with a delimiter, it is usually the case when a complex sentence starts with a subordinate clause as e.g. in the sentence *Když jsem se probudil, zavolal jsem policii.* (When I woke up, I called the police.)).

Example 12.3.
Let us illustrate the notion of a segmentation by the following sentence:

V zemi, kde každá věc má svůj řád, všechno funguje a vlaky jezdí na sekundu přesně.

[In country, where each thing has its rules, everything works and trains go on second exactly.]

(In a country in which each matter has its rules, everything works and trains go with a precision of seconds.)

[Mladá fronta DNES, 9.7.98; this sentence directly follows the sentence: *Je to ultramoderní letiště u mnohamilionové metropole technologické supervelmoci.* (It is an ultra-modern airport close to a megamillion metropolis of a technological superpower.)]

The segmentation of this sample sentence looks as follows:

W_1 : *V zemi* [In country]

D_1 : *, kde* [, where]

W_2 : *každá věc má svůj řád* [each thing has its rules]

D_2 : *,*

W_3 : *všechno funguje* [everything works]

D_3 : *a* [and]

W_4 : *vlaky jezdí na sekundu přesně* [trains go on second exactly]

D_4 : .

It is clear that the segmentation is simply a direct consequence of the division of words into the class of delimiter sequences and that of ordinary words. The application of this division to individual words immediately provides the segmentation of a sentence.

For the purpose of further processing of input sentences it is necessary to try to transform the segmentation into some kind of a proto-structure (graph) reflecting the mutual relationship of individual segments. Nodes of the graph represent the segments. Such a structure has one basic function – to identify segments which may belong to the same clause (for example if a particular delimiter is a conjunction coordinating words or groups of words) and thus to help to create a graph describing the structure of clauses in a complex sentence. The first step in this transformation is the creation of joint segments S_q by concatenation of the sequence of segments $D_iW_{i+1}D_{i+1}$ ... $W_{i+r}D_{i+r}$ (0<=r, 2(i+r)<=n), where D_i or D_{i+r} may be empty. The joint segments S_q should more precisely reflect the division of a complex sentence into individual clauses or sections of clauses (in the case of non-continuous clauses). The joint segments should be created by a special set of rules reflecting the syntactic properties of a natural language under consideration. Roughly speaking, these rules should operate on the basis of similar syntactic relations as the relations mentioned in the discussion of our motivating example above.

For a given sentence, the joint segments may be created in several ways.

Definition 12.2 (of an S-node)
An *S-node* N_q is a 6-tuple $[Q,W,V_U,V_A,D,C]$, where:

- Q is an index *q* of a particular segment S_q represented by the node;
- W is a sequence of pairs (w_i,i), where the symbol w_i represents an input symbol located on the i-th position in an input sentence and *i* is the index expressing this position. The sequence W contains only the pairs representing the lexical items belonging to the segment S_q. It is contained in curly brackets and it is called a *span of the S-node*;
- V_U is a number indicating how many symbols from W are unambiguously morphologically identified as verbs;

- V_A is a number indicating how many symbols from W have at least one tag marking a different part of speech among its tags and at least one tag marking a verb;
- D is a so called *subordination index*; this index either corresponds to the index of the node representing the segment governing the segment S_q, or it equals 0 (if no segment governs the segment S_q);
- C is a so called *index of vicinity*. It is a number corresponding to the index of the node with a lower index, to which the given segment S_q is connected by other than subordinating delimiter sequence (e.g. a coordinating conjunction etc.). In case no such node with a lower index exists, this number equals 0.

The following definition specifies how the S-nodes are organized into a graph expressing the mutual relationship between individual segments.

Definition 12.3 (of an S-graph)

An *S-graph* is a continuous acyclic graph consisting of a set of S-nodes and a set of edges. The edges are divided into the following two groups:

a) horizontal edges – these edges express the fact that the S-nodes are either coordinated or they belong to the same clause;
b) oblique edges – these edges correspond to a subordinate relation of S-nodes; the lower end of an oblique edge points towards the S-node representing a dependent segment and the upper node points towards the S-node representing the governing segment.

The set of nodes of an S-graph *St* is sufficient to fully represent *St*.

Remark

In each S-graph there is exactly one S-node with a value of D = C = 0. It is the node representing the leftmost segment of the main clause. All other nodes have at least one of these indices different from zero.

The node N_1 has the value of D = C = 0 if and only if the sentence does not start with a subordinate clause.

Let us now apply the definitions presented above to the sample sentence from Example 3:

There are the following joint segments S_q:

$S_1 = W_1$

$S_2 = D_1W_2$

$S_3 = D_2W_3$

$S_4 = D_3W_4D_4$

This sentence allows for two S-graphs:

$$\begin{tikzcd}
N_1 \arrow[r, dash] \arrow[dr, dash] & N_3 \arrow[r, dash] & N_4 \\
& N_2 &
\end{tikzcd}$$

Fig. 12.7. The first variant of the S-graph for the sentence from Example 3

N_1 stands for the node [1,{(V,1), (zemi,2)}, 0, 0, 0, 0]

N_2 stands for the node [2,{(,,3), (kde,4), (každá,5), (věc,6), (má,7), (svůj,8), (řád,9)},0,1,1,0]

N_3 stands for the node [3,{(,,10), (všechno,11), (funguje,12)}, 1, 0, 0, 1]

N_4 stands for the node[4,{(a,13), (vlaky,14), (jezdí,15), (na,16), (sekundu,17), (přesně,18), (.,19)}, 1, 0, 0, 3]

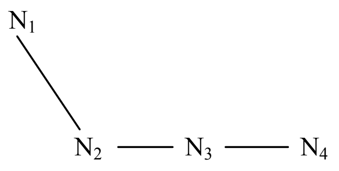

Fig. 12.8. The second variant of the S-graph for the sentence from Example 3

Nodes N_1, N_2 and N_4 are the same as in the previous case,

N_3 stands for the node [3,{(,,10), (všechno,11), (funguje,12)}, 1, 0, 0, 2]

The creation of an S-graph is only the first step towards our ultimate goal – the creation of a clause structure for a given sentence. The problem is that the coverage of individual nodes need not correspond to a set of words belonging to single clause. The coverage of a particular S-node is always continuous, while clauses may contain discontinuous elements (nonprojective constructions whose elements are located in different segments) or they may be divided into several discontinuous parts (for example when the main clause contains an embedded subordinate clause). In order to overcome this obstacle it is necessary to try to identify segments belonging to one particular clause and to join them together.

This joining of course cannot be done solely on the basis of morphological analysis of the input. One of the reasons why the morphological information is insufficient is the presence of unsafe verbs. They not only do not allow for a precise estimation of the number of clauses in a particular complex sentence, but they also substantially complicate the task of determining the ends of individual clauses in a complex sentence. Another serious problem is the ambiguity of some delimiters, especially commas. In some cases it is very difficult (if not impossible) to decide whether a certain delimiter is used in a coordinate relation inside a certain clause or whether it coordinates something else in a complex sentence. (As has been already mentioned in the point 3 of the discussion of Example 1, the comma following the name *Šavrda* in *Šavrda, úspěšný kapitán, psycholog a kouč* may have either the role of a delimiter between a noun and its attribute (in apposition) or the role of a delimiter in the coordination of nouns *Šavrda*, *kapitán*, *psycholog* and *kouč*).

A consequence of these facts is the necessity to allow for the ambiguity of structures representing joint segments. The disambiguation will then be done by later stages of parsing.

Definition 12.4 (of a CS-graph)

A *CS-graph* (a contracted *S-graph*) is a modified *S-graph.* The modification consists in a contraction of some horizontal edges of an *S-tree* and in joining nodes from both ends of the contracted edges. The nodes have the following form:

A node of a *CS-graph* N_r (a *CS-node*) is a 6-tuple [L,W,V_U,V_A,D,C], where:

- the index *r* is a list of indices of all nodes of an original *S-graph* S_q, which were joined by contraction into the node N_r;
- L is an index of the node N_r; it is the lowest index from the list of all indices from *r*;
- W is a sequence of spans of S-nodes joined into a node N_r;
- V_U is a number indicating how many symbols from W were unambiguously morphologically identified as verbs;
- V_A is a number indicating how many symbols from W have at least one tag marking a verb and at least one tag marking a different part of speech among its tags;
- D is a *subordination index*. This number corresponds to the index L of a node to which is the given node N_r connected by an oblique edge in the bottom-up direction. In case such an oblique edge does not exist, this number equals 0;
- C is an *index of vicinity*. This number corresponds to the index L of a node with a lower index to which is the given node N_r connected to by a horizontal edge. If such a node does not exist, this number equals 0.

Remark

For the sake of an easier explanation of mutual relationships of individual nodes of a CS-graph (not necessarily immediately connected by an edge) in vertical direction we introduce the notion of *layers of a CS-graph*. In informal terms, a *layer* is composed of all nodes which are mutually connected by horizontal edges only.

Definition 12.5 (of layers of a CS-graph)

Let the notion of a *layer of a CS-graph* be defined inductively according to the following steps:

1. The CS-node with D = 0 and C = 0 belongs to the *layer* 1 of the CS-graph.
2. Each CS-node connected to a CS-node belonging to a certain *layer* by a horizontal edge also belongs to the same *layer*.
3. Each CS-node N_r connected by an oblique edge to a CS-node N_p belonging to a *layer* i belongs to:
   a) the *layer* i+1 if the CS-node N_r is located on the lower end of the edge
   b) the *layer* i-1 if the CS-node N_r is located on the upper end of the edge

Remark

Let us point out that the topmost layer of a CS-graph is assigned the number 1 and the numbers of layers increase in the top-down direction.

Example 12.4.

Let us now present the *CS-graphs* corresponding to *S-graphs* from Fig. 12.7 and Fig. 12.8. The three variants of *CS-graphs* are based solely on the information about the number of safe and unsafe verbs (V_U and V_A) located in S-nodes.

a)

$$\begin{tikzcd} S_{1,3} \arrow[r, dash] \arrow[d, dash] & S_4 \\ S_2 & \end{tikzcd}$$

$S_{1,3}$ = [1, {(V,1), (zemi,2)} {(„10), (všechno,11), (funguje,12)}, 1, 0, 0, 0]

S_2 = [2, {(„3), (kde,4), (každá,5), (věc,6), (má,7), (svůj,8), (řád,9)},0,1,1,0]

S_4 = [3, {(a,13), (vlaky,14), (jezdí,15), (na,16), (sekundu,17), (přesně,18), (.,19)}, 1, 0, 0,1]

This variant is based on the *S-graph* from Fig. 12.7. It does not allow for any other kind of contraction, there is just one safe verb *funguje* (it works) in nodes S_1 and S_3 indicating that on the topmost horizontal layer the first two nodes of the original *S-graph* belong to a single clause. They are separated by an embedded clause (or a noun phrase if the ambiguous form *má* (it has/my) is a pronoun). No other contraction is possible because the node S_4 contains a safe verb and thus represents the second main clause (coordinated with the first one). The node S_2 is the only node on the bottom horizontal layer (layer 2), therefore no contraction is possible on this layer.

b)

$$\begin{tikzcd} S_1 \arrow[d, dash] & \\ S_{2,3} \arrow[r, dash] & S_4 \end{tikzcd}$$

S1 = [1, ,{(V,1), (zemi,2)}, 0, 0, 0, 0]

S2,3 = [2, {(„3), (kde,4), (každá,5), (věc,6), (má,7), (svůj,8), (řád,9)} {(„10), (všechno,11), (funguje,12)},1,0,1,0]

S4 = [3, {(a,13), (vlaky,14), (jezdí,15), (na,16), (sekundu,17), (přesně,18), (.,19)}, 1, 0, 0, 2]

This is the first of the two *CS-graphs* based on the Fig. 12.8. The ambiguous form of *má* [it has/my] is considered to be a pronoun in this variant (notice that $V_A = 0$ in $S_{2,3}$ even though $V_A = 1$ in the node S_2 of the original *S-graph*) thus allowing for the contraction of nodes S_2 and S_3 (of the original *S-graph*) into $S_{2,3}$ according to the same rule as in the previous case. The edge leading towards S_4 cannot be contracted (it contains an unambiguous verb). It is easy to see that this variant is irrelevant from the point of view of the real syntactic structure of the input sentence and that the later stages of syntactic analysis should discard it.

c)

$$\begin{tikzcd}
S_1 \arrow[dr, dash] & & & \\
& S_2 \arrow[r, dash] & S_3 \arrow[r, dash] & S_4
\end{tikzcd}$$

S_1 = [1, ,{(V,1), (zemi,2)}, 0, 0, 0, 0]

S_2 = [2, {(,,3), (kde,4), (každá,5), (věc,6), (má,7), (svůj,8), (řád,9)},1,0,1,0]

S_3 = [3, {(,,10), (všechno,11), (funguje,12)}, 1, 0, 0, 2]

S_4 = [4, {(a,13), (vlaky,14), (jezdí,15), (na,16), (sekundu,17), (přesně,18), (.,19)}, 1, 0, 0, 3]

This variant represents in fact the most adequate result; no contraction could be performed due to the decision that the word-form *má* (it has/my) is a verb (cf. the values of V_U and V_A of the nodes S_2 here and in the original *S-graph*). Thus the only node not containing a verb is the node $S_{1,}$ the only node on the topmost horizontal layer. No contraction is therefore possible on this layer.

Remark

Delimiters (commas) present in nodes created after the contraction of edges may also serve as a source of information. In some cases (the node $S_{1,3}$ in the CS-graph under a) they serve as markers indicating the location of embedded clauses.

12.4 General principles of building S-graphs

The process of building S-graphs is relatively straightforward. In accordance with the definitions presented above, the first step is always the morphological analysis of the input sentence. On the basis of its (typically ambiguous) results we divide the sentence into segments, taking into account the number and position of all delimiters and delimiter sequences in the sentence. This is quite easy, because the set of all delimiters and delimiter sequences is relatively closed and there is typically no ambiguity whether a particular word belongs to the list of delimiters or not. Joining the segments together is also a straightforward operation: the joint segments always start with a delimiter (except the first segment which need not contain any delimiter at all if the input sentence does not start with one). The last joint segment also ends with a delimiter, namely with a final full-stop, exclamation mark or question mark etc. The only slightly more complicated property of joint segments is the fact that a particular segment W_i may be empty (some examples are discussed later in this section).

The next step, drawing S-graphs relevant for a given input sentence, is slightly more complicated. A certain degree of ambiguity is involved due to those delimiters which may have both roles, the coordinating and the subordinating one. If there is such an ambiguous delimiter somewhere in the sentence, it is necessary to create two S-graphs, one with a horizontal edge and the other with oblique edge between segments connected by a particular ambiguous delimiter. The subordinating delimiters are represented by an oblique link going either up or down to the following segment (according to whether the subordinated segment precedes or follows the governing segment). There are some exceptions to this general rule. Let us explain these exceptions in the following section.

12.4.1 More complicated types of sentences

The investigations performed on the texts of the Czech National Corpus (CNC) helped us to discover one rather special type of clause structure in Czech, which is easily recognizable and requires a special treatment. Let us introduce an example:

Nevěděl, že když jsem se probral k vědomí, zavolal jsem policii.

(He_didn't_know, that when I_am Refl. woke to conscience, called I_am police)

[He didn't know that when I had gained consciousness, I had called police.]

If we look closer at this example, we will find that the underlined delimiters are not in fact a single delimiter sequence, but rather two sequences, the first one (which includes the comma) consisting of two members, the second one being just a subordinate conjunction. A very important property of this construction is the fact that both sequences of delimiters are not divided by a comma (cf. the remark following this section). The sentence consists of three segments, the first of which is the main clause of the sentence, followed by two subordinate clauses. The main problem of this sentence is to find the correct pattern how to draw a S-graph and a CS-graph, the presence of two sequences of delimiters requires a special attention. Let us demonstrate this pattern by building an S-graph.

The segmentation:

W_1 : *Nevěděl* [He didn't know]

D_1 : *, že když* [, that when]

W_2 : *jsem se probral k vědomí* [I had gained consciousness]

D_2 : *,*

W_3 : *zavolal jsem policii* [I had called police]

D_3 : .

Joint segments:

$S_1 = W_1$

$S_2 = D_1W_2$

$S_3 = D_2W_3D_3$

The S-graph created by a mechanical application of the rule that delimiters containing subordinate conjunctions are represented by an oblique edge of the S-graph would look like this:

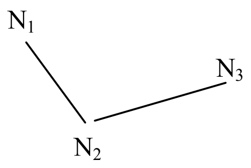

Fig. 12.9a An inappropriate S-graph reflecting the mutual relationship of segments

or, even worse, it may also look like this (if we do not take into account that there are in fact two sequences of delimiters inside D1):

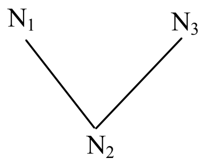

Fig. 12.9b An inappropriate S-graph reflecting the mutual relationship of segments

N_1 stands for the node [1,{(Nevěděl,1)}, 1, 0, 0, 0]

Both S-graphs from Fig. 12.9a and 12.9b differ in the vertical position of N_3. Fig. 12.9a has three vertical levels with the node N_3 placed on the middle level, while S-graph from Fig. 12.9b has only two vertical levels, with both nodes N_1 and N_3 placed on the top level.

When we try to describe the node N_2 we encounter a problem (in both cases) of setting the subordination index of the node. The S-graphs clearly show that the node N_2 depends both on N_1 and N_3 . Such a situation not only violates our definition of S-nodes (the subordination index is a single number in our definition), it also does not reflect in a correct manner the mutual relationship of clauses in our complex sentence (and that does not correspond to our goal to represent this relationship by means of S-graphs and CS-graphs). The dependency-oriented notation will definitely prefer the pattern where the third clause depends directly on the first clause and the second clause depends directly on the third one. The following S-graph would be a more appropriate representation of the clause structure than the previous two:

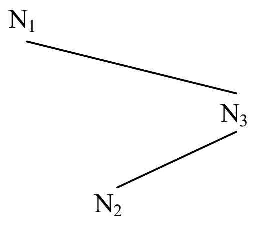

Fig. 12.10. More appropriate S-graph reflecting the mutual relationship of segments

N_1 stands for the node [1,{(Nevěděl,1)}, 1, 0, 0, 0]

N_2 stands for the node [2,{(,,2), (že,3), (když,4), (jsem,5), (se,6), (probral,7), (k,8), (vědomí,9)},1,0,3,0]

N_3 stands for the node [3,{(,,10), (zavolal,11), (jsem,12), (policii,13), (.,14)}, 1, 0, 1, 0]

This representation is in accordance with the definition of S-nodes and S-graphs. It is quite clear that this exception can be easily handled by means of rules used for creation of S-graphs. Whenever a delimiter sequence consisting of two independent delimiters (or sequences of delimiters) is encountered, the pattern of building an S-graph described above should be adopted. It is, of course, clear that in “real texts” the situation may not always be so straightforward as in our sample sentence, but our thorough investigation of texts in the Czech National Corpus brought some interesting results.

One of those results is the fact that in the whole CNC (at least in the part publicly available in the year 2000) we have not found an occurrence of three independent delimiters grouped together (although it is possible to create a Czech sentence using for example a group of delimiters *a že který když* [and that which when]). The second important fact is that although it is theoretically possible, in “real” texts there are very few examples of a clause or phrase embedded inside the segments representing both subordinate clauses. A structure of such a complexity is probably generally considered to be difficult to understand. Among the exceptions we have found rather strange sentences (*... byl jsem v tý samý situaci jako mniši jednoho kláštera, <u>který když</u> se dozvěděli, že Koperník objevil i jinačí zákony, podle kterejch se točí svět a že země tedy není centrem světa, spíš naopak, tak ti mniši páchali hromadné sebevraždy, protože si nedovedli představit jinačí svět než ten, ve kterým žili.* (... I was in the same situation as monks from one monastery, who when they found that Copernicus discovered also other rules, according to which the world turns and that the Earth therefore isn’t a center of the world, rather the other way round, so the monks committed mass suicides, because they were not able to imagine some other world than that in which they lived.)).

Remark

The work with the CNC also brought out further sentences with very similar structure to those discussed in this section. Let us consider the following sentences:

Vzpomněl si na svou ženu, které, když ji poznal, bylo také sedmnáct.

[He_remembered Refl. on his wife, whom, when her met, was also seventeen]

(He remembered his wife, who , when he met her, was also seventeen.)

Podepsal také vstupenku na vinobraní malé Michalce, která, když Miloše Zemana uviděla, tak si podle její maminky myslela, že dávají zprávy.

[He_signed also ticket on wine_festival to_little Michalka, who, when Miloš Zeman saw, so Refl. according_to her mom thought, that they_broadcast news.]

(He also signed an entry ticket to the wine festival to little Michalka, who, when she saw Miloš Zeman, thought according to her mom that they broadcast the news.)

The only difference is in the fact that in these sentences the relative pronoun *který* is preceded by a comma and it clearly depends on the main verb of the third clause, therefore the second clause is embedded into the third one and the treatment in our framework is different, more straightforward than in the previous case.

The segmentation of the sentence a):

W_1 : *Vzpomněl si na svou ženu* [He remembered his wife]

D_1 : *, které* [, which]

W_2 : {empty}

D_2 : *, když* [, when]

W_3 : *ji poznal* [he met her]

D_3 : ,

W_4 : *bylo také sedmnáct* [was also seventeen years old]

D_4 : .

Joint segments:

$S_1 = W_1$

$S_2 = D_1W_2$

$S_3 = D_2W_3$

$S_2 = D_3W_4D_4$

$$\begin{tikzcd}
N_1 & & \\
& N_2 \arrow[ul, dash] \arrow[r, dash] \arrow[dr, dash] & N_4 \\
& & N_3
\end{tikzcd}$$

Fig. 12.11. An S-graph for the sentence a)

N_1 stands for the node [1,{(Vzpomněl,1), (si,2), (na,3), (svou,4), (ženu,5)}, 1, 1, 0, 0]
N_2 stands for the node [2,{(,,6), (které,7) },1,0,1,0]

N_3 stands for the node [3,{(,,8), (když,9), (ji,10), (poznal,11)}, 1, 0, 2, 0]
N_4 stands for the node [4,{(,,12), (bylo,13), (také,14), (sedmnáct,15), (.,16)}, 1, 0, 0, 2]

Let us now contract the edges and build a CS-graph:

$$\begin{tikzcd}
S_1 & & \\
& S_{2,4} \arrow[ul, dash] \arrow[dr, dash] & \\
& & S_3
\end{tikzcd}$$

Fig. 12.12. A CS-graph for the sentence a)

S_1 = [1,{(Vzpomněl,1), (si,2), (na,3), (svou,4), (ženu,5)}, 1, 1, 0, 0]
$S_{2,4}$ = [2,{(,,6), (které,7)}{ (,,12), (bylo,13), (také,14), (sedmnáct,15), (.,16)},1,0,1,0]
S_3 = [3,{ (,,8), (když,9), (ji,10), (poznal,11)}, 1, 0, 2, 0]

12.5 General principles of building CS-graphs

Unlike in the previous case, the process of building CS-graphs on the basis of S-graphs requires much deeper and more detailed knowledge of a particular natural language. In this section we would therefore like to concentrate on general guidelines for building CS-graphs rather than on a presentation of a complete set of specific rules for a particular language. The process of gathering linguistic data and creating particular rules would require an extensive research in future, based primarily (for Czech) on the data contained both in the Prague Dependency Treebank and in the Czech National Corpus.

The discussion of the sentence structure in Example 1 at the beginning of this chapter is a good example of the type of linguistic knowledge which may be used in the process of transformation of S-graphs into CS-graphs. The most important type of information exploited in transformation rules is the information contained in S-nodes, namely the information about the number of (ambiguous and unambiguous) verbs present in a joint segment represented by the particular S-node. There are also other types of useful information, which may influence the way in which CS-graphs are created, for example the lexico-syntactic information of verbs (about the presence of reflexive particles belonging to a particular verb etc.).

The nature of rules used for transformation of S-graphs into CS-graphs is very delicate. The amount of information that we have at our disposal is not big – the (in most cases ambiguous) results of morphological analysis, lexico-syntactic information contained in the main dictionary of the system and, last but not least, S-graphs already created for a given input sentence. We should be very careful about when using this information. It will be probably very seldom the case that our linguistic knowledge supported by the data from a corpus will make it possible to formulate a general and universally valid rule. In most cases a particular rule will cover only a very specialized pattern (or configuration) present in the input sentence. In such a case it is quite clear that a huge number of rules is going to be necessary for achieving a good percentage of success. That is also a reason why we do not go further in our presentation of rules for the creation of CS-graphs – the task is simply too huge.

Let us now demonstrate some hypothesis that may serve as a basis for transformation rules:

- A very important syntactic rule in Czech says that between any two finite verbs there must be at least one sentential delimiter (if we consider complex verbal forms as one verb). This fact may be used for a whole set of transformation rules dealing with a variety of special patterns:
    - ♦ If any S-node in the a particular S-graph represents a segment containing one (safe) finite verb, all finite ambiguous verbs from the same segments are not verbs at all. This is for example the node S_1 = [1,{Vzpomněl si na svou ženu}, 1, 1, 0, 0] from our previous example, where the word form *vzpomněl* (he remembered – 3rd pers. sing. masc. or anim.) clearly indicates that the word form *ženu* (woman/I chase – verb, 1st pers. sg./ noun, acc. or loc. case, sg) is a noun, not a verb.
    - ♦ If all finite verbal forms are unambiguously recognized as verbs and if we mark their total number in a particular sentence by the letter *n* (complex verbal forms are counted as a single verb) and the total number of delimiter sequences by the letter *m,* and if $m = n - 1$, then all delimiter sequences may be considered as sentential, even if they were marked as ambiguous.
    - ♦ If there is a single delimiter between two unambiguous finite verbs then this delimiter is sentential even though it was marked as ambiguous.
- On the other hand, the fact that the number of delimiters or delimiter sequences in the sentence equals or exceeds the total number of finite verbs doesn't mean that some of these delimiters are not sentential, because the sentence may contain a verbal ellipsis. (Cf. the sentence Petr spal a Pavel také. [Petr slept and Pavel too] (Petr slept and Pavel slept, too)). It is, of course, also necessary to take into account the fact that embedded clauses sometimes require an additional delimiter marking their end, cf. the example řekl, že dozná, co udělal, a že slíbí, že… (he said that he will confess what he has done and that he will promise…).

- The reflexive particles se/si are also very important indicators (cf. the discussion of the sentence in Example 1) which segments of the sentence belong together. Due to the fact that they (together with other clitics) typically occupy the Wackernagels position and due to the high degree of word-order freedom in Czech it is often the case that a reflexive particle and the verb it belongs to are separated by one or more embedded clauses. If the sentence contains one reflexive particle in one segment and exactly one reflexive verb in another segment (or one reflexive tantum in one segment and several other verbs in other segments), it is quite clear that both segments belong to the same clause. At this point it is necessary to point out that even though the form se is ambiguous [refl. particle/preposition], in many cases it is not difficult to disambiguate it on the basis of relatively simple rules – cf. [Oliva et al. 00].
- A similar role as that of reflexive particles may be also played by individual parts of complex verb forms, for example the complex forms of the Czech past tense.

12.6 The correspondence between CS-graphs and D-trees

From the point of view of the definitions introduced in previous chapters it is interesting to find a correspondence between CS-graphs and D-trees. The key to their relationship is hidden in the data contained in the nodes of CS-graphs. According to our definition, each node of a CS-graph contains (among other) its span and the subordination and vicinity indices. The remaining information stored in each node is not relevant for the moment.

Before we start discussing the relationship of CS-graphs and De-trees we have to discuss the relationship of a CS-graph and the original input sentence. Each CS-graph (let us remind that there might be several different CS-graphs for one sentence) in fact describes one possible way how a complex input sentence may be divided into individual (hypothetical) clauses and also the mutual relationship of these clauses. Each node of a particular CS-graph (for the sake of simplicity we will use the term CS-nodes in the sequel; we will also use the term CS-edges instead of the term edges of the CS-graph) corresponds to a single (hypothetical) clause and thus the span of the node also covers the whole clause. The subordination and vicinity indices of the node in fact express the mutual positions of the clauses. It is quite easy to find a correspondence of oblique CS-edges (represented by the subordination index) and edges of the corresponding D-tree – each oblique CS-edge has its counterpart (equally oriented) in the D-tree. The edge of the D-tree leads from the head of the main clause towards the head of the subordinated clause.

The task of finding the counterpart of horizontal edges of the CS-edge in the D-tree is slightly more difficult, because the D-trees (according to the definition) do not contain any horizontal edges. The horizontal CS-edges in fact describe the coordination of hypothetical clauses. Due to the fact that the definition of D-trees does not account for any special treatment of coordination, describing the relation of coordination by means of oblique edges can be done in various ways. It is up to the author of the metagrammar to decide whether the governing node of the coordination is the first or the last member of this relation or whether the coordinating conjunction should be the governing node of the coordinated group. Accordingly, the horizontal CS-edges are then transformed into edges of the D-tree, capturing the relationship between heads of coordinated clauses in a way chosen by the author of the particular metagrammar.

The correspondence between CS-graphs and D-trees leads to a very important consequence for parsing. The fact that each CS-node corresponds to one (hypothetical) clause and that the CS-edge have direct counterparts among the edges of the corresponding D-tree (connecting heads of these clauses) allows for an independent parsing of these clauses. If we replace CS-edges by corresponding edges of a D-tree and if we replace the CS-nodes by subtrees obtained as a result of parsing the spans of CS-nodes, we will get a De-tree covering the whole sentence. If parsing of a clause covered by a particular CS-node fails, it is a signal that this CS-graph does not represent a correct clause structure of the sentence and that it should be discarded.

Remark

These considerations hold for projective sentences and for sentences containing non-projective constructions which do not exceed the span of one clause. At first sight this might seem inadequate, but from the practical point of view we can hardly expect that any practical system (not using the analysis of clause complexity by means of CS-graphs) would be able to analyze more complicated non-projective constructions (for example the construction, where the element constituting the non-projectivity is extracted from the subordinate clause into the main clause). As for Czech as a model language of our approach, we may raise serious doubts about the grammaticality of nonprojective constructions exceeding the span of one clause in declarative sentences (except a single type of constructions similar to the sentence *Krásnou říkal, že si Petr vzal ženu* [Beautiful he_said, that Refl. Petr married woman] (He said that Petr married a beautiful woman)).

12.7 The connection between CS-graphs and our metagrammar

It was already mentioned that one of the main reasons for the construction of CS-graphs is our endeavor to reduce the complexity of the analysis by filtering out the subtrees which are acceptable only in a local context, but not globally in the context of the whole input sentence. The derivation of superfluous structures should be prevented as soon as possible, they are an unnecessary burden having a detrimental effect on parsing efficiency. Let us now show one possible way of a direct application of CS-graphs of a particular input sentence in the process of its analysis and thus (at least partially) fulfil our goal.

We have already defined what we understand under the term *layer of the CS-graph* in previous paragraphs. There is quite a straightforward way in which the numbers expressing these layers might be used in our metagrammar. More precisely there are several ways of applying the layers of CS-graphs. Let us now present at least the most obvious ones:

- Let us suppose that all items representing words from the span of a particular CS-node will receive a special attribute *layer* with the value equal to the value of a layer of that particular CS-node. This value may be used for blocking the creation of superfluous items, which would put together items belonging to different layers. The block may be implemented in the relevant metarules for example in the form of the following IF statement:
  IF A.layer = B.layer THEN ELSE FAIL ENDIF
  This statement would be very useful for example during the parsing of complex sentences with embedded clauses. If there is an embedded clause (which may govern other subordinate or even embedded clauses to make things more complicated), then this statement allows to analyze the main clause divided by an embedded clause in a correct manner, not allowing to combine subtrees belonging to different clauses.
  This method of using the information about layers has some drawbacks, for example, it does not guarantee the proper attachment of subtrees in case there are two or more clauses in the same layer. On the other hand, it is very simple and easily applicable.
- Slightly more complicated is the second method of application of *layers*. It exploits the assumption that individual CS-nodes are very likely to represent individual clauses of the input sentence. The rules according to which the CS-graphs are created should be written with this assumption in mind. This fact allows for even more refined use of information contained in CS-graphs. In this case we are going to introduce a special attribute *index* into all items representing words from spans of a particular CS-node. The value of this attribute will equal L (the index of the particular CS-node). This value is going to be used precisely for the same purpose and in the same manner as the attribute *layer* in the previous case.
  The main advantage of using the attribute *index* over using the attribute *layer* consists in solving the problem of two clauses being in the same layer, on the other hand this method is more vulnerable to errors which may appear during the creation of CS-graphs.
- The third method, requiring deeper modifications of the parsing algorithm, exploits both types of information mentioned in previous paragraphs, namely the information about the *index* and *layer* of a particular CS-node. This method is based on the same assumptions as the previous two methods, and exploits the fact that if we have a CS-graph for the whole sentence, we may start the analysis from most deeply embedded subordinate clauses (represented by CS-nodes on the layer with the highest number). When the analysis of all nodes from one layer is finished, it continues

on the immediate higher layer until it reaches the top layer (and analysis the main clause(s)) of the sentence.
It is quite clear that this method would make it possible to replace a complicated task of the analysis of a complex sentence into a much simpler task of the analysis of a series of simple clauses and thus it may radically speed up the analysis. It has the same drawback as the previous method, namely the vulnerability to ill-formed CS-graphs.

All three methods presented above were described as if there were only a single CS-graph for a particular input sentence. This is, of course, not very likely to happen, a good deal of ambiguity is going to be involved in the process of building the CS-graphs. We may suppose that the most likely outcome (at least for more complicated sentences) will be a set of CS-graphs for each complex input sentence instead of just a single graph.

This fact will, of course, influence all three methods presented above. It will probably have the smallest influence on the first one, because even if there are two or more distinct CS-graphs for a particular input sentence, individual words may belong to the same layer in all cases. In other cases it will probably be necessary to parse the input sentence several times, each time with different values of the attribute *index* for individual input items.

This fact may have direct consequences especially for the second method which will probably need a very sophisticated control system which will take care of reusing the items already computed during the analysis of other CS-graphs for a particular complex sentence.

This need not necessarily be the case of the third method where it may be much easier to reuse the subtrees representing individual clauses – if a particular clause represented by a particular CS-node belongs to different layer or is connected with different CS-node in some other CS-graph, it should not be necessary to repeat its analysis. In this case we should be able to reuse a lot of already existing items. Anyway, it is clear that all these considerations are only tentative and that it will be necessary to choose the “proper” method of exploitation of CS-graphs only after the proposed method of building CS-graphs is implemented and reliable experimental results are available.

Conclusion

The main task of this dissertation was to describe the problem of creating a robust parser for a language with high degree of word-order freedom from both basic aspects – the theoretical and implementational one. In the theoretical part we have continued the work already presented in several papers ([Holan et al. 98], [Holan et al. 00], [Plátek et al. 01]), in the practical part we have developed a sample (meta)grammar of Czech which served not only as a testing tool for ideas described in the theoretical part, but it also inspired additional theoretical research reflecting (and trying to cope with) the problems encountered during the development, testing and debugging of the metagrammar.

As we have already mentioned, our work directly followed up with the research in the following theoretical fields:

- We have created (in cooperation with Martin Plátek, Tomáš Holan and later also with Karel Oliva) a theoretical framework capable to describe in an adequate manner both the syntactically ill-formed and non-projective constructions. As we have shown in our papers in the past, the problems of non-projectivity and ill-formedness are very closely related.
- We have developed (in cooperation with the team mentioned above) a set of measures providing a basis for formulation of global constraints restricting the application of the metagrammar and thus providing more adequate set of results of parsing by filtering out unacceptable results.

Let us now briefly summarize the results described in this dissertation:

- We have adopted the theoretical basis described above for the implementation of a particular metagrammar of Czech. We have used the measures of non-projectivity and robustness in that implementation and thus we have demonstrated that the theoretical background may serve as a cornerstone of more practically oriented work.
- We have practically demonstrated in our metagrammar that due to the relaxation of global constraints applied on interpretation of individual metarules it is possible to capture quite complicated concepts of non-projectivity and robustness by relatively simple metarules. This is possible thanks to the fact that one metarule is interpreted in a different manner according to which constraints are relaxed either globally, in a given moment, or locally, for one particular metarule.
- The experience with our metagrammar also influenced (and changed) our original assumptions concerning the magnitude of non-projectivity for Czech. We have originally estimated that it equals one, but our experiments with the sentences from the testbed inspired a deeper investigation of this phenomenon which finally lead to a new, more adequate hypothesis – cf. [Holan et al. 00].
- We have formulated an estimation of the influence of the degree of non-projectivity on parsing complexity showing that there is a direct correspondence between these two phenomena which can be expressed in quite a precise manner.
- We have implemented a metagrammar handling a wide range of syntactic phenomena of the language and allowing to demonstrate several problems of robust parsing of languages with a high degree of word-order freedom.
- We have developed a method for testing and debugging a large-scale metagrammar and a set of sentences (testbed) used by this method both for preserving the consistency of the metagrammar in the process of its incremental development and for demonstrating of advantages and disadvantages of our approach.
- One of the most important results of our work is contained in the last chapter. The theoretical framework described there provides a basis for a system which might have a substantial influence on parsing complexity and adequacy of parsing for complex sentences. This framework is of course only a first step towards a more adequate handling of complex sentences, but it clearly

shows that the work presented in this dissertation was not wasted. It opens a new path leading towards our ultimate goal of an adequate and efficient robust parser of Czech.

- Last but not least result of our work concerns the methodology of organizing the research of a very complex problem involving several scientific fields. Even though every member of our team was mainly engaged in solving the problems of his particular field, we have been able to influence and inspire each other. From the point of view of the author of the metagrammar, the development of a large scale metagrammar still remains a one man job, even though our approach allows a distribution of work at least in the sense that the metarules can be created to a large extent independently from the investigation and specification of global constraints. The close cooperation with both author of the interpreter and the author of the underlying theory definitely makes this task more simple.

References

[Abney 96] Abney, S: Partial parsing via finite-state cascades. In Proceedings of the ESSLI'96 Robust Parsing Workshop, Prague, 1996.

[Bémová, Kuboň 90] Bémová, A., Kuboň, V.: Czech-to-Russian Transducing Dictionary, In: Proceedings of the 13th International Conference COLING, Vol. 3, pp. 314 – 316, University of Helsinki, 1990

[Bémová 96] Bémová, A.: Analysis of Nominal Groups, JRP PECO 2824 Language Technologies for Slavic Languages, Final Research Report, Appendix I., Prague, 1996

[Bémová et al. 97] Bémová, A., et al.: Anotace na analytické rovině, ÚFAL Technical Report TR-1997-03, Charles University Prague, 1997

[Brants 99] Brants, T.: Cascaded Markov Models; in: Proceedings of EACL'99, University of Bergen, 1999

[Ciravegna, Lavelli 99] Ciravegna, F., Lavelli, A.: Full Text Parsing using Cascades of Rules: An Information Extraction Procedure; in: Proceedings of EACL'99, University of Bergen, 1999

[Colmeraurer] Colmerauer, A.: Les Systèmes Q ou un formalisme pour analyser et synthetiser des phrases sur ordinateur. Mimeo, without date, Montreal

[Gladkij 73] Gladkij, A.V.: Formal'nyje gramatiky i jazyki, Iz. Nauka, Moskva, 1973

[Hajič 94] Hajič, J.: Unification Morphology Grammar, PhD thesis, MFF UK, Praha 1994

[Hajič 98] Hajič, J. et al.: Core Natural Language Processing Technology Applicable to Multiple Languages, in: Final report of the Workshop'98 of the Center for Language and Speech Processing at the Johns Hopkins University, Baltimore, 1998

[Hajič, Hladká 98] Hajič, J. Hladká, B.: Tagging Inflective Languages. Prediction of Morphological Categories for a Rich, Structured Tagset. ACL-Coling'98, Montreal, Canada, August 1998, pp. 483-490.

[Hajičová 95] Hajičová, E. (Ed.): Text-and-Inference Based Approach to Question Answering, Theoretical and Computational Linguistics, Vol.3, Fac.of Philosophy, Charles University, Praha 1995

[Hajičová et al. 99] Hajičová, E., Panevová, J., Sgall, P.: Manuál pro tektogramatické značkování, ÚFAL Technical Report TR-1999-07, MFF UK Praha, 1999

[Holan et al. 97] Holan, T., Kuboň, V., Plátek, M.: A Prototype of a Grammar Checker for Czech, In: Proceedings of the Firth Conference on Applied Natural Language Processing, pp.147-154, Washington, DC, March 1997

[Holan et al. 98] Holan, T., Kuboň, V., Oliva, K., Plátek, M.: Two Useful Measures of Word Order Complexity, In: Proceedings of the Workshop Processing of Dependency-Based Grammars, COLING-ACL'98, Universite de Montreal, 1998

[Holan et al. 00] Holan, T., Kuboň, V., Plátek, M. and Oliva, K.: On Complexity of Word Order, In: Les grammaires de dépendance - Traitement automatique des langues Vol 41, No 1 (2000) (special issue on dependency grammars of the journal Traitement automatique des langues, guest editor Sylvain Kahane).

[Holan 01] Holan, T.: Nástroj pro vývoj závislostních analyzátorů přirozených jazyků s volným slovosledem, (A Software Environment for the Development of Dependency Parsers for Natural Languages with Free Word Order) (in Czech),Disertační práce, MFF UK, Praha, 2001

[Kirschner 87] Kirschner, Z., APAC3-2: An English-to-Czech Machine Translation System, In: Explizite Beschreibung der Sprache und automatische Textbearbeitung XIII, Prague 1987

[Kirschner 94] Kirschner, Z.: CZECKER - a Maquette Grammar-Checker for Czech, The Prague Bulletin of Mathematical Lingistics 62, MFF UK Prague, 1994, pp. 5 - 30

[Kuboň et al. 97] Kuboň, V., Holan, T., Plátek, M.: A Grammar Checker for Czech, ÚFAL Technical Report TR-1997-02, MFF UK Praha, 1997

[Kunze 72] Kunze, J.: Die Auslassbarkeit von Satzteilen bei koordinativen Verbindungen im Deutschen, Berlin: Akademie-Verlag, 1972

[Kunze 75] Kunze, J.: Abhängigskeitsgrammatik, Berlin: Akademie-Verlag, 1975

[Marcus 65] Marcus, S.: Sur la notion de projectivité, in Zeitschrift für mathematische Logik und Grundlagen der Mathematik XI, 1965, pp. 181-192.

[Nebeský 72] Nebeský, L.: A Projectivity Theorem, in Prague Studies in Mathematical Linguistics, 3, Academia, Praha, pp.165-169

[Oliva 89] Oliva, K.: A Parser for Czech Implemented in Systems Q, in Explizite Beschreibung der Sprache und automatische Textbearbeitung, MFF UK Praha, 1989

[Oliva 93] Oliva, K.: StriPTtTTT: String Processor & Text-to-Tree(s) Transformation Tool, JRP PECO 2824 Language Technologies for Slavic Languages, Research Report, Saarbruecken, 1993

[Oliva 96] Oliva, K.: A Grammar Checker for Czech, JRP PECO 2824 Language Technologies for Slavic Languages, Final Research Report, Prague, 1996

[Oliva et al. 00] Oliva, K., Hnátková, M., Petkevič, V. and Květoň, P.: The Linguistic Basis of a Rule-Based Tagger of Czech, In: TSD 2000, Proceedings (eds. Sojka, Kopeček, Pala) Lecture Notes in Artificial Intelligence Vol. ,pp.3-8, Springer, 2000

[Panevová 80] Panevová, J., Formy a funkce ve stavbě české věty (Forms and Functions in the Syntax of Czech Sentences - in Czech), Academia, 1980

[Petkevič 95] Avgustinova, T., Bémová, A., Hajičová, E., Oliva, K., Panevová, J., Petkevič, V.(ed.), Sgall, P. and Skoumalová, H.: Linguistic Problems of Czech. Final Research Report for the JRP PECO 2824 project. Prague, 1995

[Plátek et al. 01] Plátek, M., Holan, T., Kuboň, V.: On Relax-ability of Word-Order by D-grammars, Tech. Report TR-2001-01, MFF UK, Praha, 2001, Accepted at DMTCS 2001 conference.

[Sgall et al. 86] Sgall, P., Hajičová, E., Panevová, J., The meaning of the sentence in its pragmatic aspects, Mey, J.L., ed., Reidel, 1986.

[Sikkel 97] Sikkel, N.: Parsing Schemata - A Framework for Specification and Analysis of Parsing Algorithms, Texts in Theoretical Computer Science - An EATCS Series, ISBN 3-540-61650-0, Springer-Verlag Berlin/Heidelberg/New York, 1997

[Sikkel, Nijholt 97] Sikkel, N., Nijholt, A.: Parsing of Context-Free Languages, in: Handbook of Formal Languages, G.Rosenberg and A.Salomaa (eds.), Springer, Berlin and Heidelberg, 1997

[Skoumalová 94] Skoumalová, H.: Czech Dictionary for the Grammar Checker, JRP PECO 2824 Language Technologies for Slavic Languages, Research Report, pp 130-139, Saarbruecken, 1994

[Panevová, Straňáková 99] Panevová, J., Straňáková, M.: Some Types of Syntactic Ambiguity; How to Treat them in an Automatic Procedure. In: TSD'99, Proceedings (eds. V. Matoušek, P. Mautner, J. Ocelíková, P. Sojka), Lecture Notes in Artificial Intelligence vol.1692, Springer, pp. 50-55, 1999.

[Starý 97] Starý, J.: Statistický model syntaktických vztahů v povrchovém zápisu české věty, (Statistical Model of Syntactic Relations in the Surface Description of a Czech Sentence), in Czech, Diplomová práce na MFF UK, Praha 1997

[Straňáková 99] Straňáková, M.: Selected Types of Pg-Ambiguity. The Prague Bulletin of Mathematical Llinguistics 72, Praha, pp. 29-58. 1999

[Straňáková 01] Straňáková, M.:Homonymie předložkových skupin a možnost jejího automatického zpracování. (The Homonymy of Prepositional Groups and the Possibility of Their Automatic Processing), in Czech, Disertační práce, MFF UK, Praha, 2001