

MASARYK UNIVERSITY
FACULTY OF INFORMATICS



Enhancing Czech Parsing with Complex Valency Frames

MASTER THESIS

Miloš Jakubíček

Brno, 2010

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Advisor: RNDr. Aleš Horák, Ph.D.

Acknowledgement

First of all I would like to thank my advisor Aleš Horák for all the patience with me, for encouraging and motivating leadership, and great collaboration in the past three years. It was a wonderful opportunity for me and I learnt a lot in the community of our NLP Centre where I was glad to meet with you and Jan Bušta, Marek Grác, Dana Hlaváčková, Vojtěch Kovář, Karel Pala, Pavel Rychlý, and others – thank you all for your invaluable help, neighbourliness and willingness.

Special thanks go to Vladimír Kadlec, former developer of `Synt`, for his goodwill to explain whatever I missed in the 40,000 lines of C code.

Last but not least my thanks go to my family for that they steadily support their son.

Abstract

In this thesis an exploitation of the verb valency lexicon Verbalex for the Czech parsing system *Synt* is presented and an effective implementation is provided that uses the syntactic information in the complex valency frames contained in Verbalex to resolve standard parsing ambiguities and thereby improving the analysis results. A notion of positive and negative valencies is introduced and several customizations how they can be employed in the parsing process are offered. Finally, an evaluation is shown that uses the test suite for *Synt* and the Brno Phrasal Treebank.

Keywords

syntactic analysis, parsing, verb valency, complex valency frames, natural language processing, Synt, Verbalex

Contents

1	Introduction	2
1.1	<i>NLP: A Brief Survey</i>	2
1.2	<i>Syntax and Parsing Technology</i>	3
1.3	<i>Exploitation of Valencies in Parsing</i>	5
2	Synt Parser	7
2.1	<i>Parsing Workflow</i>	8
2.2	<i>Grammar</i>	8
2.3	<i>Facing Ambiguity in Parsing</i>	13
2.4	<i>Analysis Output</i>	16
3	Verbalex Lexicon	17
3.1	<i>Complex Valency Frames</i>	17
4	Exploiting Valencies	19
4.1	<i>From Synt to Verbalex</i>	19
4.2	<i>From Verbalex to Synt</i>	20
4.3	<i>Ranking Valency Hypotheses</i>	22
4.4	<i>Positive and Negative Valencies</i>	23
5	Implementation	26
5.1	<i>Preparation of the Verbalex Data</i>	26
5.2	<i>Searching for Valencies</i>	28
5.2.1	<i>Triggering the Search Action</i>	29
5.2.2	<i>Search Algorithm</i>	30
5.2.3	<i>Matching and Scoring</i>	30
6	Evaluation	32
6.1	<i>The Brno Phrasal Treebank</i>	32
6.2	<i>Test Suite for Synt</i>	32
6.3	<i>LAA Tree Similarity Metric</i>	35
6.4	<i>Evaluation of Valency Contribution</i>	35
6.4.1	<i>Overall Performance</i>	35
6.4.2	<i>Positive valencies</i>	36
6.4.3	<i>Negative valencies</i>	36
6.5	<i>A Critical Note on Parsing Evaluation</i>	37
7	Conclusion	38

Chapter 1

Introduction

“Language is a part of our organism and no less complicated than it.”

Ludwig Wittgenstein

1.1 NLP: A Brief Survey

The topic of this thesis falls within the field of natural language processing (NLP), a well-established research domain at the interface between computer science and linguistics, which however bears a variety of other names besides NLP: it has been on one hand referred to as computational (or mathematical or formal) linguistics, emphasizing the linguistic part of the domain; and on the other hand as language technology (or engineering), emphasizing the computational workmanship. A not unusual notion is also human language processing (HLP), though (as I see language as one of the key characteristics of humankind) I consider it being a near-synonym to NLP. In this paper I’m going to use all of those terms interchangeably, all of them denoting a field with bold long-term goals:

- **theoretical**
formal understanding and modelling of natural language, including its analysis, synthesis, acquisition, reasoning and other cognitive processes
- **applicational**
building of useful (computer) applications solving practical tasks in text and speech processing, either defined as analysis and synthesis on particular language level (phonetics, morphology, syntax, semantics,

pragmatics and logic – being an extra-language level) or as specific applications across several of levels (information retrieval and extraction, machine translation, text production, dialog systems, computational lexicography etc.)

It should be noted that these two approaches also more or less correspond to the duality of this domain and are not always consistent with each other: while the theoretical (linguistic) part has basically grounded this research field in 1960s (benefiting from research of formal languages), the applicational aspects became more and more important with increasing computing power and current development is predominantly driven by the applicational approach ¹. However I'm quite confident that one should strive for keeping both sides of this field in a (frangible but still) balance with the common aim to build a well-formalized, computationally efficient and functional theory of language which will serve as a useful basis for practical high-quality applications.

1.2 Syntax and Parsing Technology

The goal of parsing is usually defined as “recovering the structure of sentence”, but this (or any other) vague definition can never really fit. This is one of the notable differences to the analysis and processing of morphology and possibly one of the reasons why morphology can be ticked off as solved for several well-resourced languages, while – to my best opinion – syntax (and any other higher level) processing is far away from a similar status in any language, English being of course counted as the most-developed one. In no way this means that morphology would be an easy task in general – there are vast inter-language differences², but in certain aspects morphology is significantly easier to solve than syntax: while morphology can be seen as a clearly defined task with well-measurable solutions and high inter-annotator agreement when creating relevant language resources manually, processing of syntax suffers from several difficulties:

- As mentioned above, the task itself can be formalized only up to a certain level of abstraction. In the past there have been several (mutually

1. The current status quo was nicely formulated by Shuly Wintner: “Are we applied statisticians now?” [Win10] I do believe we should not be.

2. At this point I cannot resist quoting the head of our department, professor Karel Pala, who occasionally irritates English native speakers saying that English morphology is a toy work compared to the Czech (and many other languages as well of course) one.

incompatible) syntax theories developed, altogether basically providing evidence that there is nothing like *the* syntax and *the* correct parse of a sentence, a fact I'm strongly convinced of³. Hereby I also state that this work does not comply to any specific syntax theory but focuses on a general improvement of a particular tool (a parser) by supplying additional information (valency structure).

- The implication of the previous paragraph is that syntax must be (again, on contrary to morphology) strongly application-driven. There is not a single syntax (*per sé, an sich*) and different applications require different kinds of syntactical information stored in efficient structures (by efficiency I mean not only the implementation side, but also an "efficient content"). Building useful tools for syntax processing that are generally applicable is an extremely complex (and limited) task. A nice conclusion of this issue has been given by Harry Bunt in [BT96, p. 2]: "Parsing is usually not a goal in itself, but is intended to be instrumental to the recognition, interpretation, or transformation of complex structures. The relative importance of the issues relating to parsing technology therefore depends very much on the intended application".
- The fact that even domain professionals achieve insufficient inter-annotator agreement when dealing with syntax provides additional supporting evidence for the two previous statements and considerably affects the quality of manually built resources for syntax as well as subsequent evaluation which is based on them – this problem is discussed in Section 6.5 in detail.
- There is strong evidence (see e. g. [Cro92] for more on this topic) that people use extra-syntactical information to resolve standard parsing ambiguities. Since most of current semantic resources suffer from various deficiencies (low coverage being the most common one), high ambiguity still remains to be one of the most burning problems in parsing.

Apart from the standard phrase-structure and dependency formalisms, there have been several other complex frameworks for syntax developed, such as Head-Driven Phrase-Structure Grammar (HPSG, [PS94]), Lexical Functional Grammar (LFG, [Bre01]), Tree-Adjoining Grammar (TAG, [JS97]),

3. Nevertheless I would be pleased by studies within the area of computational psycholinguistics and neurolinguistics that would disprove it.

Combinatory Categorical Grammar (CCG, [Ste00]) and others. Though some of them (especially HPSG and LFG as syntactic-semantical systems) try to achieve very high coverage of various syntactic phenomena, their applicability might be questionable as well as related parsing complexity issues. A notable example of an opposite direction in development, a simplification of parsing process, is the SET parsing system [KHJ09], currently being also developed in the NLP Centre at Masaryk University in Brno.

A last remark on parsing issues I would like to postulate is that one question seems extremely appealing to me: is the “parsing problem” actually not *overdefined* and *overspecified*? Are trees (or graphs, or similar non-flat structures) really the right representation for syntax?⁴ This and many other theoretical questions (e.g. how much information is indeed “stored” in syntax) still remain to be answered.

1.3 Exploitation of Valencies in Parsing

In [Hau01, s. 303] the following key combination principles in syntax are given (all of which have the general left-associative derivation order in common):

- Valency
- Agreement
- Word Order

While morphological agreement and word order can be seen a dual principles (languages with weak agreement, e.g. English, possess fixed word order and vice versa, e.g. Czech), verb valency⁵ represents a common principle and basically stands for a bridge between syntax and semantics. In further text we use the notion of the *valency frame* (also called *subcategorization frame*) which consists of a given verb and its valency positions, i.e. gaps which need to be filled with arguments (also called *fillers*). Hence a general valency frame schema of arity n looks as follows:

$$\text{verb}(\text{argument}_1, \text{argument}_2, \dots, \text{argument}_n)$$

A verb frame instantiated with corresponding arguments is also called a *functor-argument* or *predicate-argument* structure. Crucially a verb cannot

4. Don't we want more than we can actually have?

5. In further text we focus only on verbs as primary valency carriers, though other part of speech (like e.g. prepositions) might be accompanied with a valency structure as well.

be combined with its arguments arbitrarily, but imposes strict requirements on what combinations of word forms represent valid argument structure for the verb. This can be for a given language L (a set of all word forms), $V \subset L$ (a proper subset of all verb lemmas) and $n \in \mathbb{N}$ (valency frame arity) formalized using a constraint function c :

$$c : V \times L^n \rightarrow \{0, 1\}$$

$$c(v, (w_1, w_2, \dots, w_n)) = \begin{cases} 1 & \text{if } (w_1, w_2, \dots, w_n) \text{ is a valid instantiation} \\ & \text{of a valency frame of verb } v \\ 0 & \text{otherwise} \end{cases}$$

A set F_v of all valid valency frames for a given verb v can be then formulated as:⁶

$$F_v = \bigcup_{i=1}^{\infty} \{p \in L^i \mid c(v, p) = 1\}$$

Not unlike many other NLP problems, while valency frame can be formalized trivially, it is not easy to provide valuation of this function (i. e. to create a verb valency lexicon) for a particular language. Opinions about what is and what is not a valency of a given verb differ heavily, since in many cases the boundary between an obligatory argument (valency, complement) and an optional modifier (adjunct) is rather fuzzy.

The exploitation of verb valencies in parsing has been theoretically studied in [Ně06] and a prototypical implementation has been given in [HHK06]. In this work we focus on a practical implementation in the Czech parser Synt using the verb valencies obtained from the valency lexicon Verbalex [HH06] and evaluate the impact of using valencies on parsing quality.

6. Of course the infinity in the formula is just a formalization eye-candy – practically, i tends to be less than 4.

Chapter 2

Synt Parser

Syntactic parser *Synt* [Kad07] has been developed for several years in the Natural Language Processing Centre at Faculty of Informatics, Masaryk University. It is based on a context-free backbone, performs a stochastic agenda-based head-driven chart analysis using the provided grammar for Czech, employs several disambiguating techniques and offers both a phrase-structure and dependency output.

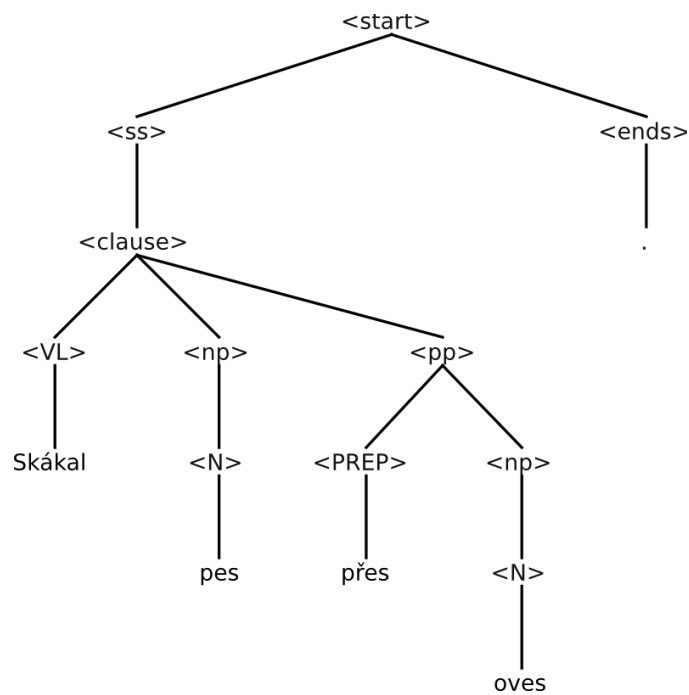


Figure 2.1: A sample phrase-structure tree from *Synt* output

2.1 Parsing Workflow

An overview of the parsing workflow of the `Synt` parser is given on Figure 2.2. First, the input sentence is morphologically analyzed if the annotation is not provided along with the sentence (the parser accepts ambiguous morphological annotation which might be even significantly refined as a side-effect of the analysis – refer to [JHK09] for more details).

As the next step a basic head-driven CFG analysis is performed. All parsing results are collected in the so called *chart* structure which is built up during the analysis. Chart is a multigraph where each node corresponds to a word and edges represent (possibly partially) matching grammar rules (see example on Figure 2.6). A key feature of this approach is that while chart itself has polynomial size (with respect to the length of the input sentence), it can store all, i. e. up to exponential number of analyses of the given sentence. Therefore chart represents a structure suitable for further effective post-processing of the analysis result.

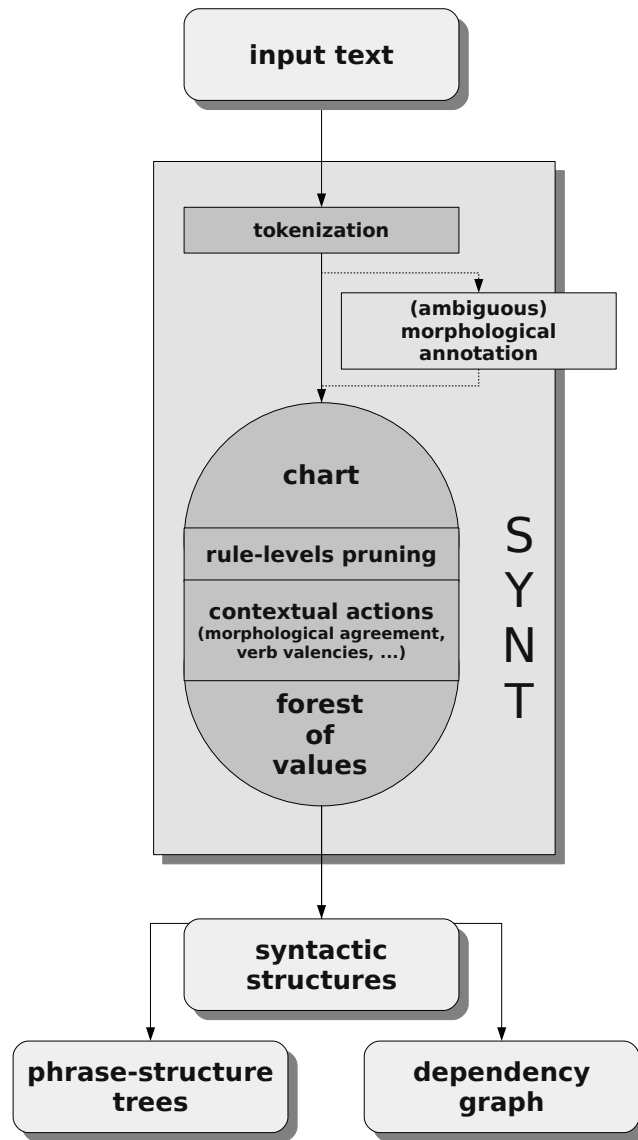
Upon the basic CFG analysis, we build a new graph structure called *forest of values*¹. This structure originates from applying contextual actions on the resulting chart and from the implementation point of view it comes more handy for post-processing than the original chart (while it still keeps the polynomial size) – a sample forest of values is given on Figure 2.7: nodes (values) in one row build a value node list which represents children within a single analysis. One value may have links to multiple value node lists, i. e. multiple alternative analyses.

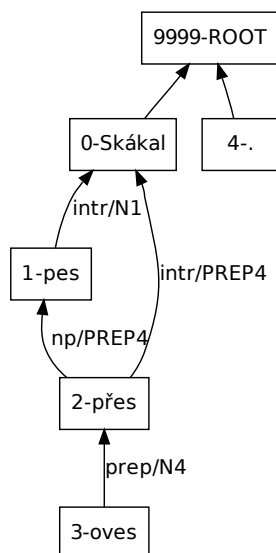
Finally, `Synt` offers three output possibilities from the resulting forest of values: a phrase-structure tree (Figure 2.1), a dependency graph (Figure 2.3) or a set of syntactic structures (Figure 2.4).

2.2 Grammar

To prevent maintenance problems which a hand-written grammar is susceptible to, the grammar used in `Synt` is edited and developed in the form of a small meta-grammar (described in detail in [KH05]) which at this time contains only 239 rules. From this meta-grammar a full grammar can be automatically derived (having 3,867 expanded rules). In addition, each rule can be associated with several attributes – a brief description of an example rule is given on Figure 2.5:

1. The name is actually misleading and I keep using it just to be compatible with former work – a more fitting notion would be e. g. a *value graph*.

Figure 2.2: Parsing workflow of the *Synt* parser

Figure 2.3: A sample dependency graph from the *Synt* output

[0-9) : Muž , , vede kolo
 man-N lead-V bike-N
(a man leads a bike)

[2-6) : který stojí u cesty
 who-PRON stands-V at-P road-N
(who stands at the road)

Figure 2.4: A sample result of extraction of structures using the *Synt* parser for the input sentence: "Muž, který stojí u cesty, vede kolo." (*A man who stands at the road leads a bike.*)

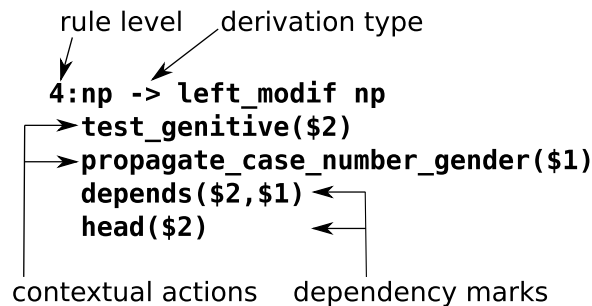


Figure 2.5: A meta-grammar rule example

- a set of one or more so called **actions**, i. e. programmed functions that are called when the given rule matches and may modify the analysis result (adjust parse ranking or even prune an analysis) and are mostly used for covering contextual phenomena such as grammatical agreement,
- a **derivation template** which is used to derive the full grammar and which can define various operations on the rule which are used to obtain additional alternations, such as permutation of the right-hand side of the rule, inserting a particular non-terminal between each two non-terminals on the right-hand side or expanding each non-terminal on the right-hand side to its own right-hand side in the grammar (i. e. sort of “skipping” the non-terminal),
- a **rule level** which allows stratification of the grammar into separate levels of structural complexity which can be enabled or disabled at runtime and make it possible to improve the parsing precision or add domain specific rules (where by domain we refer e. g. to uncommon text like tables, mathematical formulas and similar “chunks” of strings) into the grammar that are employed only if they are necessary to parse the sentence (as described in Section 2.3),
- a **head or dependency marker** which are a used to build a dependency graph from the chart analysis.

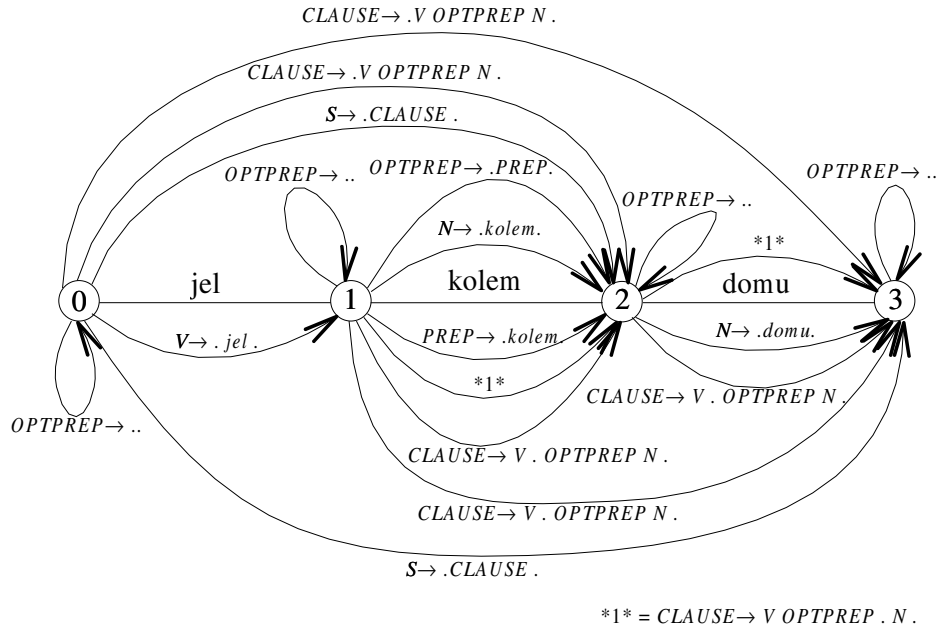


Figure 2.6: A sample chart structure (example taken from [Hor07])

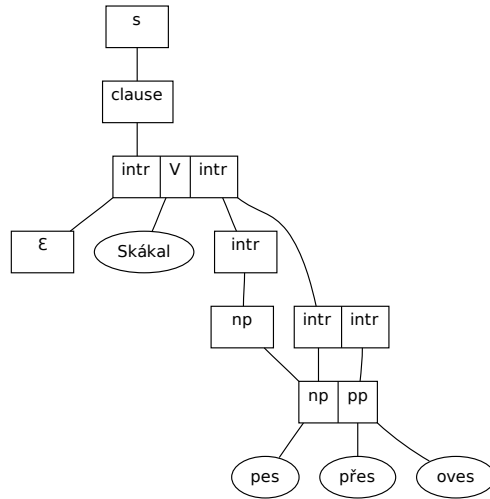


Figure 2.7: A sample packed shared forest. Note that for visualization purposes this is a slightly simplified version.

2.3 Facing Ambiguity in Parsing

In recent evaluation [Kad07, p. 77] it has been shown that `Synt` accomplishes a very good coverage (above 90 %) but the analysis is highly ambiguous: for some sentences even billions of output syntactic trees can be given. Currently, there are several techniques employed in `Synt` to fight the ambiguity that are further extended in this work by exploiting verb valencies as described in Chapter 5:

- **Morphological Agreement**

As mentioned in Section 1.3, morphological agreement plays a key role in parsing of fleective free-word-order languages such as Czech and therefore it is used as one of the main instruments for reducing ambiguity of the analysis in `Synt`.

Each rule can be assigned a set of morphological tests (described in Section 2.2) that can detect particular grammatical features or agreement between members of the right-hand side of a rule (realized as an edge in the chart).

All these contextual actions are triggered after the basic CFG analysis and serve as a basis for creating the forest of values.

- **Ranking and Selection of *n*-best Trees**

Probabilistic parsing (or statistical NLP in general) represents as such a large domain which is very popular and under intensive research. `Synt` performs a standard probabilistic CFG (also referred to as PCFG) analysis using information of rule probabilities in the grammar. Those probabilities have been estimated from the Brno Phrasal Treebank [KJ08] which is described in Section 6.2 in detail.

The ranking of a single tree is calculated as the product of all rule probabilities that are used in the tree and can be furthermore modified by all contextual actions. Finally, an efficient selection of *n*-best trees according to this ranking is possible (see [HSK02]).

- **Pruning according to rule levels**

The stratification of the grammar into separate rule levels (see 2.2) can be used not only to be able to easily enable and disable a set of rules, but also to prune unnecessarily complex (and hence improbable) analyses after the parsing.

This is a very handy feature, since we often cannot predict the structural complexity of the text in advance, i. e. whether we are going to parse a text of common sentences or rather ungrammatical chunks of strings that might not exceed the level of noun (or other) phrases in their structure. In such situations we can enable all rule levels by default, parse the input, determine the lowest rule level that builds a successful analysis and prune all analyses that use rules from higher levels.

This feature can actually be seen as dual to the probability ranking – while probabilities are observed in corpus, rule levels represent a hand-written priority and complexity of rules which we can directly influence and significantly improve the coverage of parsing without harming the precision (increasing ambiguity).

- **Chart Beautifyization**

When developing an effective and compact context-free grammar for natural language, it is hard to avoid the usage of many rather technical non-terminals, i. e. such that are not really linguistically motivated but help to keep the grammar small and robust.

An obvious disadvantage is that those are later also present in the resulting trees – they don't mind, but make the analysis result notably less readable and may also contribute to unnecessary parsing ambiguity.

Therefore `Synt` has been enhanced by a sophisticated chart-based beautification algorithm [KH07] that efficiently cleans the analysis from “unimportant” non-terminals. Moreover, analyses that become identical after the beautifyization can be merged into a single one.² An example showing the effect of this procedure is provided on Figure 2.8.

- **Optimal Maximum Coverage**

Besides using a well-developed grammar stratified into many rule levels (which might be difficult to create as well as maintain), there is another approach for covering ungrammatical chunks of text, usually called *partial parsing* or *chunking*. A similar feature has been implemented in `Synt` called *Optimal Maximum Coverage* [AKRC05] – it can

2. To be more precise: only one of those analyses is kept and all others can be pruned. Formally, given A_G be the set of all possible analyses based on non-terminals from grammar G and $b : A \rightarrow A$ the beautifyization function, the beautifyization algorithm computes the quotient set A/ρ where $\forall a_1, a_2 \in A : a_1 \rho a_2 \iff b(a_1) = b(a_2)$.

be triggered upon an unsuccessful analysis and compute an optimal combination of partial analyses into a single complete analysis.

Although this may look like a technique useful for increasing coverage only, it is mentioned here to draw the attention to the fact that it might not be desirable to develop an all-covering grammar that will necessarily suffer from huge ambiguity problems. Using methods like Optimal Maximum Coverage allows us to keep the grammar strict and nevertheless parse texts that simply don't represent well-formed sentences.

2.4 Analysis Output

Synt possesses three basic output possibilities, all of which are created from the forest of values and available as command-line output and (in case of PS-trees and dependency graph) via a web interface³ [Gol07]:

- **Phrase-Structure Trees**

The main output of Synt is usually a set of phrase-structure trees ordered according to the tree ranking, two example trees have been given on Figure 2.8.

- **Dependency Graph**

A dependency graph represents a packed structure which can be utilized to extract all possible dependency trees (see example provided on Figure 2.3). It is created by using the head and dependency markers that might be tied with each rule in the grammar.

- **Syntactic Structures**

Finally, the third output option of Synt is realized by means of so called *extraction of structures* (described in more detail in [JHK09]) – an unambiguous optimal decomposition of the input sentence into a set of syntactic structures that are required by the user. An example of extraction of clauses is given on Figure 2.4.

3. At <http://nlp.fi.muni.cz/projekty/wwwsynt/>

Chapter 3

Verbalex Lexicon

The Verbalex valency lexicon for Czech has been continuously developed since 2004 [HH06] in the NLP Centre at Masaryk University. Currently it contains over 21,000 verb frames for more than 10,500 Czech verbs. It uses the notion of two-level annotation, a syntactic and semantical one, in the so called *complex valency frames* [PH08]. On the first level, “shallow” syntactic valencies (direct grammatical cases, prepositions with cases etc.) are available, whereas the second level contains deep semantic annotation using *semantic roles*. Moreover, each valency frame is not bound to a single verb, but to a synonymic set (*synset*) mapped to the Czech WordNet [PS04].

3.1 Complex Valency Frames

The internal representation of Verbalex uses both an XML and plain text format. An illustrative example valency frame for the verb “skákat” (to jump) is provided on Figure 3.1. Its structure is as follows:

- on the first line, the related WordNet synset is provided – all verbs in the synset share the information given below,
- the second line contains the frame itself: a first-level semantic role of all arguments is denoted by upper-case letters (AG – agent, LOC – location, ENT – entity), each of which is accompanied by a list of second-level roles that are again mapped to the WordNet (by their

```
[1] přehoupnout se1, přehupovat se1, přešvihnout se1, přeskakovat1, přeskočit1, skákat2, skočit2 ≈  
-frame: AG<person:1|animal:1>obl VERBobl LOC<location:1>|ENT<stream:1>obl  
-example: skákal přes příkop (impf)  
-example: kůň přeskočil přes potok (pf)  
-synonym: přehoupnout1, přehupovat1  
-use: fig (přehoupnout se, přehupovat se); prim (přešvihnout se, přeskakovat, přeskočit, skákat, skočit)  
-reflexivity: no (přeskakovat, přeskočit, skákat, skočit); refl (přehoupnout se, přehupovat se, přešvihnout se)
```

Figure 3.1: Example valency frame for the Czech verb “skákat” (to jump)

meaning number). Finally, all valency “gaps” end with the syntactic information – in the example with direct case (“kdo1” – animate nominative) and a prepositional case specification (“přes+co4” – over + inanimate accusative), each of which can be either obligatory (marked as OBL) or optional (OPT),

- following lines contain example sentences, a list of synonyms (that do not share the same valency frame), subclassification on the type of usage (primary, figurative or idiomatic) and reflexivity of the verbs.

Chapter 4

Exploiting Valencies

4.1 From Synt to Verbalex

The close relation between valencies in Verbalex and parsing by Synt has been as far studied in [JKH09], where we used the extraction of syntactic structures for obtaining a lower-bound estimation of Verbalex coverage on the Czech corpus DESAM [PRS97] as well as supplying missing verbs and their valency frame candidates including relevant examples.

The basic idea was as follows: the extraction of structures in Synt has been extended so that we were able to construct shallow valency frames for each verb that we then compared to the valency frames available in the Verbalex lexicon. The exact extraction procedure was as follows:

1. identify clauses in the input sentence and process each of them separately,
2. in each clause, identify all prepositional and noun phrases, infinitives and selected conjunctions recorded in Verbalex: “až” (till), “že” (that), “jestli” or “zda” (whether), “ať” (let be), “aby” (in order to), “jak” (how),
3. construct a valency frame using the above information together with available morphological annotation,
4. for all automatically extracted valencies of a particular verb, check whether they are available in Verbalex.

We observed that Verbalex is a valuable language resource that achieves a very high coverage on common Czech texts – both verb and valency coverage exceeded 80 %.

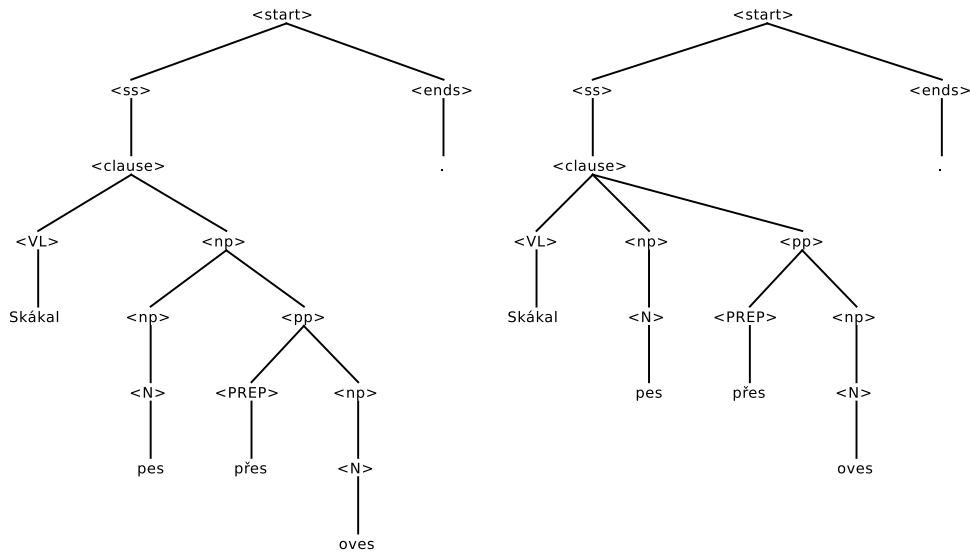


Figure 4.1: A PP-attachment ambiguity example.

4.2 From Verbalex to Synt

The work described in this thesis goes into opposite direction: how can the Verbalex lexicon contribute to the analysis performed by Synt?

As was briefly outlined in Section 1.3, valency frames carry very important syntactic information in that they distinguish obligatory arguments of a verb from its optional modifiers. This kind of information has a (theoretically straightforward) exploitation in parsing for resolving various attachment ambiguities.

On Figure 4.1, a standard PP-attachment ambiguity issue is shown. Without additional information there is no way for the parser to know what analysis should be the preferred (or even right) one. The preposition “přes” (over) can introduce an argument (as presented on the figure), an adjunct or even a part of a noun-phrase argument as is illustrated in Table 4.1 – all of this depends mainly on the particular verb.

However if we could efficiently lookup the valency frame of the verb “skákat” (which has been demonstrated on Figure 3.1) and provide its content to the parsing process, we would be able to determine the right (or at least more probable) hypothesis and prefer it over other analyses found in the preceding parsing steps. This can be basically done in two ways:

Argument
Skákal pes přes oves jump-V dog-N over-P oat-N The dog jumped over the oat.
Adjunct
Položil tužku přes papír put-V pen-N over-P paper-N He put the pen over the paper.
Noun-phrase part
Sednul si s nohou přes nohu. sat-V oneself-PRON with-P leg-N over-P leg-N He sat down and crossed his legs.

Table 4.1: An overview of possible syntactic positions of a prepositional phrase.

- **pruning unsatisfied hypotheses**

This is an aggressive but straightforward way: all parses that fulfill some selection (e. g. that they do not fully match any of the valency frame for the given verb) will be removed from the set of possible analyses, effectively reducing number of resulting parse trees. Naturally the main disadvantage is that we might prune correct hypotheses that may not match any valency frame for several reasons:

- because of an error in the valency lexicon (and, since manually built language resources often suffer from consistency problems, this must be taken into consideration as a standard situation),
- because of an elliptical construction which can cause one or more arguments of the verb to be missing within the scope of the clause,
- and unfortunately also due to an error in parsing which might have unpredictable consequences.

- **altering ranking of unsatisfied hypotheses**

Altering the parse ranking on one hand doesn't reduce the ambiguity of the analysis, but on the other hand it is not susceptible to the possible overpruning effect described above and, moreover, if combined with e.g. selection of n -best trees, it might effectively lead to the same results.

However it immediately introduces the burning question how the current ranking (based on edge probabilities) should be modified in order to account for the valency frames reasonably. A first necessary step that needs to be done is that all hypotheses will be ranked according to how much they satisfy valency frames of the particular verb – an algorithm to solve this problem is proposed in the next section.

4.3 Ranking Valency Hypotheses

A valency structure found in a parse hypothesis might differ from the valency frame as given in the lexicon in that it contains more or less arguments and/or that some of the arguments do not satisfy the restrictions imposed by the valency frame.

Moreover, several hypotheses may completely match different valency frames.¹ In such cases we would like to prefer the hypothesis that covers the largest (with respect to the number of arguments) valency frame.

Therefore, our ranking algorithm (which presumes that we have already found all valency structures – hypotheses – in the sentence) proceeds as follows:

1. Let V be the set of all found valency structures and F the set of all valency frames obtained from the lexicon for the particular verb. For each $v \in V, f \in F$, *found* be the number of all found arguments, *correct* the number of them which are satisfied in the valency frame and *size* be the number of all arguments in the frame, we define the scoring function s :

$$s : V \times F \rightarrow \mathbb{R}$$

given as:

1. This typically occurs for verbs with monovalent nominative valency as well as e.g. a divalent nominative–accusative (or any other case) one.

$$s(v, f) = \begin{cases} size & \text{if } found = correct = size \\ \frac{2 \cdot precision \cdot recall}{precision + recall} & \text{otherwise} \end{cases}$$

where

$$precision = correct / found$$

$$recall = correct / size$$

The overall score s_v of a found valency structure $v \in V$ is then defined as:

$$s_v = \max_{f \in F} (s(v, f))$$

Basically, we compute the maximum f_1 measure of the found valency structure and given valency frames and among those structures that achieve $f_1 = 1$, we prefer larger structures.

2. As next step, we may multiply the probability of the analysis by the score of the associated valency structure or set a valency score threshold and prune all analyses that achieve lower score.

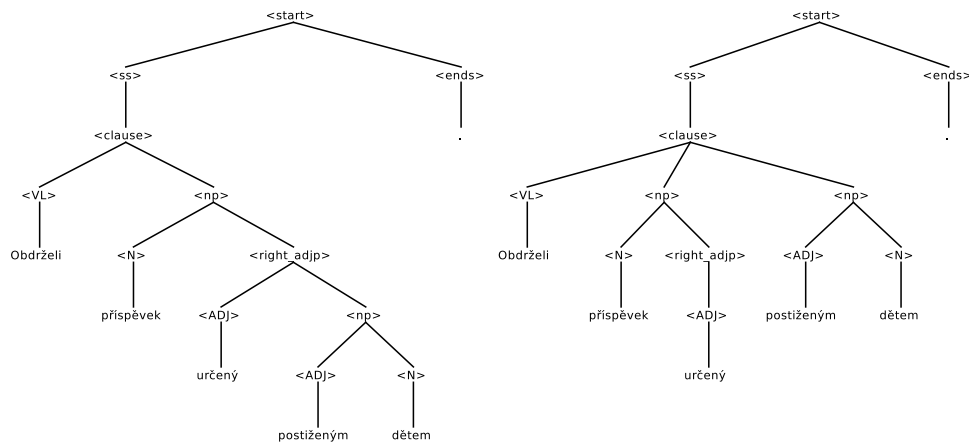
4.4 Positive and Negative Valencies

In this section the notion of *positive valencies* and *negative valencies* is introduced. Valencies in the way they have been exploited in the previous section represent positive valencies that should be matched in order to positively contribute to the ranking of all analyses. Of course they can also contribute negatively, but the purpose of the scoring function as defined before is to alter the current ranking of all analyses in a way that will (to some extent) respect their valency structure score.

There might be however certain situations in which we would like to explicitly disprefer a hypothesis, regardless of its score that would compute our function s . In such cases we can use what I will refer to as negative valencies. They basically represent a list of forbidden adjuncts that are not able or are very unlikely to be combined with the given verb.

An example sentence with negative valency is presented on Figure 4.2 – since the transitive verb “obdržet” (to obtain, to receive) does not possess

4. EXPLOITING VALENCIES



∅ Obdrželi příspěvek určený postiženým dětem
 They-N obtain-V contribution-N intended-A disabled-A children-N
 They have obtained a~contribution intended for disabled children.

Figure 4.2: An example sentence with ambiguity solvable by negative valencies.

a dative valency, the analysis shown on the right side of the figure is clearly wrong and we can disprefer or prune it.

An obvious problem comes up with this approach how to determine the situations when we can safely apply a negative valency rule. This information is unfortunately not present in the Verbalex lexicon, therefore we proceed with the following (language dependent) heuristics for Czech:

- No **prepositional valencies** can be handled as negative valencies, since all of them are able to occur in the position of an adjunct as adverbs indicating time, manner, place, degree, frequency etc.:

– Koupil počítač v obchodě.
 he-N-Nom buy-V-Past computer-N-Acc in-P shop-N-Loc
 He bought a computer in the shop.

- For the same reason **instrumental** (7th case) is out of the scope of negative valencies as it is heavily used as an adverb of manner:

– Rychlým krokem vešel dovnitř.
 he-N-Nom quick-A-Ins pace-N-Ins enter-V-Past inside-Adv
 He entered quickly.

- **Local** (6th case) is used with prepositions only and hence cannot be handled as well (see example for prepositional valencies above).
- **Vocal** (5th case, used to address somebody or personalized something) is not considered to be a valency of a verb and therefore it always takes the position of an adjunct (denoting the addressee). Besides it is quite rare in written text.
- **Dative** *usually* does not take the position of an adjunct, but there are notable examples when it does and is used to express a patient or a benefitor:

Dojdi otci pro pivo.
 go-V-Cmd father-N-Dat for-P-Acc beer-N-Acc
 Bring a beer for your father.

Note that the grammatical agreement is between “pro” and “pivo” in Czech, i. e. the preposition “for” does not take the same semantics as in English here.

- Finally, **nominative, genitive and accusative** are of great interest with respect to negative valencies since none of those cases can be attached to a verb as an adjunct.

It must be emphasized that the classification given above is in no way complete and there are exceptions (in both directions) since the ability of a verb to bind an adjunct is determined also (and mainly) by semantics. Therefore it would be of great benefit to have such information in the Verbalex lexicon: a negative enumeration of forbidden adjuncts can be seen as complementary information to the current content of Verbalex (a positive list of arguments and some adjuncts).

Chapter 5

Implementation

In this chapter a brief description of an efficient implementation is provided based on the procedures and algorithms described in previous parts. Since the `Synt` parser is implemented in the C programming language, all of this work has been also brought in as a separate C module of the `Synt` system (except for the converter of Verbalex data) that can be enabled/disabled at runtime using the command-line option `-vf <valency_file>` and offers several customization possibilities.

In the first section, the preparation and format of the data from the Verbalex lexicon is described, followed by an explanation of algorithm used for retrieving valency structures from (partial) analyses and completed by the description of the matching and scoring procedures.

5.1 Preparation of the Verbalex Data

Verbalex lexicon represents a very large database that needs to be converted into a comprehensive, easily readable (for a program) format that will contain only the information that is indeed used during the analysis. The lexicon is primarily stored in an XML format with a plain text counterpart which we use to convert from. An example synset valency frame from the plain text Verbalex format is provided on Figure 5.1.

As for now, only the syntactic information from Verbalex is used to contribute to the analysis in `Synt`, therefore the information we actually need to extract from each frame is rather limited. The converter itself (file `verbalex2synt.py`) has been written in Python and quickly (in less than 3 seconds even on low-end laptop¹) processes the Verbalex plain text input file to what I call *inverted valency format* which can be efficiently read and stored in memory by `Synt`.

The format itself is very simple: it is line based, each line starting with a (unique) valency frame, followed by a comma-delimited list of verbs that

1. The average over 10 runs measured by the `system time` utility was 2.8665 s.

```

* PŘESKOČIT:5, SKOČIT:7
=wnid: CZE30-36880902-v
=wnilr type=hypernym: ENG20-00092462-v
=def: rychle jako skokem se dostat
=canbepassive: no
=meaning: I
=class:
  ~ dok: přeskočit:5 dok: skočit:7
  + INS(col;<device:1>;obl)+++VERB+++TIME(na co4;<time unit:1>;obl)
    -synon:
    -example: dok: hodiny skočily na osmou
    -occurs: not-verbalex
    -attr: use=prim, reflexivity=no, mustbeimperative=no

```

Figure 5.1: An example of a plain text valency frame for the synset “PŘESKOČIT:5,SKOČIT:7” (to jump, to jump over sb./st.) from the Verbalex lexicon.

```
1-7za|3ke-2z,přistěhovat se,imigrovat
```

Figure 5.2: An example line in the inverted valency format for the perfect verb “přistěhovat se” (to move in) and biaspectual verb “imigrovat” (to immigrate).

share this valency frame. The valency frame itself is reduced on a hyphen-delimited list of valencies, each of which is either a number (denoting a direct case) or a number followed by a preposition name (denoting a prepositional case) or a list of alternatives of both prepositional and direct cases (delimited by a vertical line). The file created by the converter from current Verbalex lexicon has the length of only 352,125 bytes. An example line in the inverted valency format is shown on Figure 5.2.

The inverted valency format is read by `Synt` on startup and effectively stored in the memory into a hash containing the verb name as key and a list of possible valency frames as value. Using this approach all the strings (verb names as well as valency frames) are stored in the memory only once, all together requiring a space of only 1768 kB.² Thanks to the hashing, the valency frame list can be looked up in constant time for a given verb. The valency hash is allocated only at the beginning in case of batched processing of multiple sentences (though reading the file and allocating the hash takes only very little time – 0.0351 s³).

2. Containing 10,549 verbs and 29,068 verb-frame pairs, on a Linux machine with page size of 4 kB.

3. The time is an average over 10 runs when starting `Synt` on empty input has been

5.2 Searching for Valencies

The search algorithm itself is run on the forest of values structure as described in Section 2.1. Although the forest of values represents an effective and compact packing of all analyses into a polynomial structure, its processing and subsequent gathering information is not trivial – a naïve exhaustive search would still lead to exponential explosion and is hence unfeasible.

The valency frames are looked up within the scope of a single clause (triggering the algorithm is described in detail in the next section) and we need to retrieve all possible valency structures (“frames candidates”) within the relevant clause. To achieve this, we employ a dynamic technique on a depth-first search of forest of values. The algorithm itself starts top-down (from the clause edge), but a recursive implementation of depth-first search allows us to collect the results bottom-up.

What are actually the valencies that we try to find? For our purposes, valencies are values (in the forest of values) that correspond to a noun-phrase or prepositional phrase edge in the original chart structure. Note that we do not necessarily need to search the whole forest of values to find all valencies: basically, we follow all descendants of the clause edge to find the first noun- or prepositional phrase on the way – after that there is no need to continue the search.

However, there are several caveats that need to be taken into consideration:

- Since we cannot perform an exhaustive search, we must not visit a value (in the graph – forest of values) twice and need to remember partial results – valencies that have been already found by a previous visit of this value,
- The results collected for separate analyses (i. e. separate value node lists) must be merged to the parent value,
- The results collected *within a single analysis* for separate children values (i. e. children of a value within a particular tree) must be combined to the parent value while the dynamic algorithm must ensure that the valencies of a particular value remain independent for recurrent visits of the same node. Note that this step still suffers from possible combinatoric explosion but the number of valencies is usually small and hence it does not represent a problem in practice.

measured with and without loading valencies (using the system utility `time`).

```

/* pattern: a (possibly reflexive) infinitive
   preceded by another verb */
%group verbAI={
    vi_list:verb_rule_schema($@, "(#1)")
        groupflag($1, "head")
        test_valencies($@, "$1")
        depends($1, intr),
    R vri_list:verb_rule_schema($@, "(#2 #1)")
        groupflag($2, "head")
        depends($2, $1)
        depends($2, intr)
        test_valencies($@, "$2 $1")
}

/* pattern: I have to read */
clause %> vo_list group(verbAI)
    depends(getgroupflag($2, "head"), $1)
    verb_rule_schema($@, "#1")

```

Figure 5.3: The upper example shows how valency actions (`test_valencies`) are bound to a group of verb rules (infinitives preceded by another verb), while the second one display a rule where the valency action will be applied automatically (since it is tied to the `group(verbAI)` rule group). The action takes all right-hand side values of the rule (`$@`) as the first argument and a string containing a pointer to the relevant verb and possibly also to the reflexive pronouns “se, si” as the second one (“`$2 $1`”).

5.2.1 Triggering the Search Action

Searching for valencies, matching and subsequent reranking must occur when all the necessary information is available (i. e. the analysis has been completed up to the clause level) but we can still influent the ranking of all hypotheses (i. e. it is not a post-processing step). Therefore triggering the valency actions has been implemented as a contextual action (as described in 2.2) in the grammar, tied to particular `clause` rules.

Hereat the design and expressional power of the meta-grammar formalism used in `Synt` was proven to be an outstanding concept. Also, since `clause` rules constitute the core of the grammar, they are carefully designed and achieve high coverage what concerns possible verb(s) structure of a Czech sentence within a single clause.

Two example modifications of the grammar to trigger valency actions are presented on Figure 5.3.

Algorithm 1 A pseudo-code of the depth-first search of forest of values

```

Require: child
result =  $\emptyset$ 
for all analysis  $\in$  analyses do
  for all child  $\in$  children of analysis do
    if !child $\rightarrow$ hasBeenVisited then
      child $\rightarrow$ hasBeenVisited = 1
      if isNounPhrase or isPrepPhrase then
        child $\rightarrow$ valencies = createValency()
      else
        child $\rightarrow$ valencies = dfsSearch(child)
      end if
    end if
    analysis $\rightarrow$ valencies = combine(analysis $\rightarrow$ valencies, child $\rightarrow$ valencies)
  end for
  if multipleAnalysis then
    setValue(analysis $\rightarrow$ valencies)
  end if
  result = merge(result, analysis $\rightarrow$ valencies)
end for
return result

```

5.2.2 Search Algorithm

A pseudo-code of the search algorithm is given as Algorithm 1. It shows the `dfsSearch` function that is run recursively on the forest of values and dynamically collects possible valency frames. Note the `setValue` function that stores for each found valency frame a pointer to the value node list (there might be several of them) where the analysis diverted from another one. This is necessary to be able to modify the ranking or prune the hypothesis after all valency frames are calculated.

5.2.3 Matching and Scoring

After the search algorithm, the matching and scoring procedures are run on resulting valency structures. For each structure its scoring functions s is computed and afterwards negative valencies are evaluated. It is possible to customize the aggressivity of the valencies at runtime by the `-va <number>` command-line option to `Synt`, which can take following values:

- 0 **dry-run** mode: no ranking modifications and no pruning is performed, just the valency search is performed and its output is displayed,
- 1 – 4 **normal** mode: analyses ranking is multiplied by the value of the scoring function s and the given value,
- 5 **aggressive** mode: the ranking of all analyses with score function less than 1 is set 0,
- 6 **pruning** mode: all analyses with score function less than 1 are pruned.

Chapter 6

Evaluation

6.1 The Brno Phrasal Treebank

The Brno Phrasal Treebank (BPT) was created in years 2006–2008 as a product of linguist specialists collaborating with the NLP Centre at Masaryk University and is still under continuous development.

Currently the corpus contains in overall 86,058 tokens and 6,162 syntactically annotated sentences. The main source of the sentences is the Prague Dependency Treebank [Haj98](which may allow future comparisons on parallel sentences), which cannot be used directly because it uses the dependency formalism and there is also some evidence that it suffers from severe consistency problems – this has been studied by Vojtěch Kovář and me in [KJ09].

The key property of BPT is that it is created not just as a treebank, i. e. set of trees, but as a corpus of *syntactic limits*, restrictions that a phrase-structure parse tree must fulfill in order to be considered as correct. Using these limits a parse tree for the treebank is selected from the output of `Synt` (if it is present).

Although this procedure might look strange and suspicious at first sight, it is a very good prevention from overfitting to the treebank. Moreover, it can help us avoid consistency problems that treebanks (like any other manually built language resources) are susceptible to [KJ09].

6.2 Test Suite for `Synt`

Developing a parser and a hand-written grammar is a demanding and time-consuming task even for domain professionals. Therefore it is of great benefit to have a tool which can be used for evaluating the impact of intended changes immediately.

Such a tool has been developed for the `Synt` parser as a joint work of Vojtěch Kovář and me in 2008 and published in [KJ08]. It is based on the Brno Phrasal Treebank and performs an automatic run of morphologically

annotated data from the treebank through the `Synt` parser, compares the results of the analysis with the treebank and collects important statistical information (a sample output is shown on Figure 6.1):

- **overall number of parse trees**
Used for estimating grammar precision/ambiguity – 3 values are actually recorded: number of sentences that have not been accepted (0 parse trees), the median and the mean number of parse trees.
- **number of trees in limits**
Number of trees that fulfill the limits as given in BPT, this number tells us more or less how many “correct” trees have been found in the output. Ideally there should be only one; if there are no such trees, the output of the parser is incorrect (the grammar undergenerates); if there are more of them, the limits recorded in the treebank should be probably more restrictive (but it may also indicate a parser error). These two values are therefore recorded separately (number of sentences with less than one tree in limits, number of sentences with more than one tree in limits).
- **parser stability**
Number of sentences that caused the parser to fail (in some way) or have been timeouted is monitored as well.
- **similarity**
Similarity of the parsing results to the “gold” trees in the treebank using LAA similarity metric as described below. We keep track of 3 values: the best score achieved among the first 100 output trees, the average score (among the same set) and the score of the first (best-ranked) tree.

The first version of the test suite suffered however from a few issues, the most burning one being the duration of the test which was almost 4 hours even on high-end machines. This was caused by the way how the test had to be run: for each sentence, a separate parser process was started twice, once for the analysis with limits from the treebank and once without them.

The fact that the `Synt` process was spawned each time anew was the main issue that caused such a long time to be needed for a test – especially memory allocation and deallocation is a very costly operation which was performed over and over again.

BASIC TEST RESULTS

```

Sentences                : 5917
Passed limits             : 5859 (99.02 %)
More trees in limits     : 2194 (37.08 %)
Not accepted              : 0 (0.00 %)
Failed                   : 0 (0.00 %)
Timed out                : 58 (0.98 %)
Median number of trees   : 72
Average number of trees  : 1774709.42
Average LAA (best LAA value) : 82.25
Average LAA (first 100 trees) : 77.63
Average LAA (first tree) : 78.73
Time left                : 0:44:15
Time per sentence        : 0:00:00.448707

```

Figure 6.1: Sample output of a test suite run.

Step	Overall time	Avg./sentence
First version	3 h 41 min	2.15 s
Merging analysis with/without limits	1 h 23 min	0.80 s
Batched processing (with LAA)	43 min	0.42 s
Batched processing (without LAA)	14 min	0.14 s

Table 6.1: Summarization of the test suite speed up.

As part of the work on exploiting valencies, both `Synt` and the test suite have been customized to significantly speed up the tests. The results of this development are provided in Table 6.1: First, `Synt` was modified to allow performing both an analysis with and without limits within a single run.¹ This step has already shorten the duration of the test to about a half. Second, the test suite was modified in order to allow batched processing of all sentences in `Synt`, i. e. the process was spawned only once (daemon-like). This reduced the test time by another 70 % so that the current bottleneck is already the LAA comparison (which is run on the first 100 trees).

1. Note that by single run I mean the run of the process, not of the analysis that still needs to be performed twice.

6.3 LAA Tree Similarity Metric

As a similarity metric between a “gold” tree in the treebank and a parse tree from the output of the parser, we use a metric called *leaf-ancestor assessment* (LAA) [Sam00] proposed by Geoffrey Sampson in 2000. We consider this metric to be more reliable than the older PARSEVAL metric which is however (and unfortunately) still used more frequently.

The LAA metric is based on comparing so called *lineages* of two trees. A *lineage* is basically a sequence of non-terminals found on the path from a root of the derivation tree to a particular leaf. For each leaf in the tree, the lineage is extracted from the candidate parse as well as from the gold tree. Then, the edit distance of each pair of lineages is measured and a score between 0 and 1 is obtained. The mean similarity of all lineages in the sentence represents the score of the whole analysis.

In [SB03], it is argued that the LAA metric is much closer to human intuition about the parse correctness than other metrics, especially PARSEVAL. It is shown that the LAA metric lacks several significant limitations described also in [BSDH98], especially that it does not penalize wrong bracketing so much and it is not so tightly related to the degree of the structural detail of the parsing results.

In the test suite, an implementation of the LAA metric by Derrick Higgins is used.²

6.4 Evaluation of Valency Contribution

Using the test suite for *Synt* several evaluations have been performed on the Brno Phrasal Treebank that are shown below. Especially the scalability of positive valencies provides promising results, however finding the right valency aggressivity value might differ for various input data.

6.4.1 Overall Performance

In Table 6.2, an overall performance of the valency algorithm is demonstrated. It can be seen that the average number of retrieved valencies is relatively low (8.2) and the search procedure does not significantly worsen time performance of the parser.

2. Which is publicly available at <http://www.grsampson.net/Resources.html>

run	avg VF	max VF	time
without valencies	–	–	0.147 s
with valencies	8.2	947	0.162 s

Table 6.2: A time comparison of two test suite runs – values in the second and fourth column are an average per sentence.

–va	LAABest	LAAFirst	LAAAvg	trees
0	82.25	78.73	77.63	72
1	82.25	78.73	77.69	72
2	82.25	77.97	77.85	72
3	82.45	79.73	78.13	72
4	84.12	79.76	78.17	72
5	84.12	79.76	69.18	72
6	84.12	75.02	64.63	38

Table 6.3: A performance overview of positive valencies – first column is the valency aggressivity as given by –va.

6.4.2 Positive valencies

In Table 6.3 an overview of positive valency performance is provided – as we can see the performance was best for the valency aggressivity 4 setting, where we achieved about 2 % better LAA best score.

The performance drop for valency aggressivity 5 and 6 (especially for the LAA average value) might be explained by the fact that some of the first 100 analyses have been negatively modified or pruned respectively.

6.4.3 Negative valencies

The negative valencies as shown in Table 6.4 have provided positive but only little effect. The reason for this was that they occur quite rarely and most probably even less rarely within the first 100 trees.

type	LAABest	LAAFirst	LAAAvg	trees
without valencies	82.25	78.73	77.63	72
negative pruning	82.25	78.73	77.83	65

Table 6.4: A performance overview of negative valencies (nominative, genitive and accusative have been tested).

6.5 A Critical Note on Parsing Evaluation

As was already mentioned, even if big effort is made to eliminate annotator errors, some of them will still remain – and we dare to predict that the number of errors grows with the size of the data in a non-linear way. This raises a question whether treebanks actually represent a good way for measuring parser quality at all. Besides treebank consistency issues, there is more evidence which makes the use of treebanks for parser evaluation questionable: often the evaluation is significantly influenced by the different formalisms, annotations and last (but definitely not least!) by different linguistic insights and opinions. It should be mentioned that there have been recently proposed application-driven evaluation techniques for parsers (see e. g. [MSS⁺09]) which we believe to continue becoming more widely used in the parsing community.

Hereat I may return to raising the same questions this short work in syntax started with, because I do believe that the issues with evaluating parsers are closely related to the question what actually the goal of parsing should be.

Chapter 7

Conclusion

In this thesis an exploitation of the verb valency lexicon Verbalex in the Czech parsing system *Synt* has been presented, including an efficient implementation for searching and matching valencies in common texts. The notion of positive and negative valencies has been introduced and the evaluation shows that both of them can positively contribute to the parsing procedure.

Future development may focus on exploiting semantic information available in Verbalex and use the valency frames a bridge between syntax and semantics.

Bibliography

- [AKRC05] Marita Ailomaa, Vladimír Kadlec, Martin Rajman, and Jean-Cédric Chappelier. Robust stochastic parsing: Comparing and combining two approaches for processing extra-grammatical sentences. In Stefan Werner, editor, *Proceedings of the 15th NODALIDA Conference, Joensuu 2005*, pages 1–7, Joensuu, 2005. Ling@JoY.
- [Bre01] Joan Bresnan. *Lexical-functional syntax*. Wiley-Blackwell, 2001.
- [BSDH98] Srinivas Bangalore, Anoop Sarkar, Christine Doran, and Beth Ann Hockey. Grammar & parser evaluation in the XTAG project, 1998. <http://www.cs.sfu.ca/~anoop/papers/pdf/eval-final.pdf>.
- [BT96] Harry Bunt and Masaru Tomita. *Recent Advances in Parsing Technology*, chapter 1. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.
- [Cro92] Matthew Walter Crocker. *A logical model of competence and performance in the human sentence processor*. PhD thesis, University of Edinburgh, 1992.
- [Gol07] Jiří Golembiovský. *WWW rozhraní k syntaktickému analyzátoru SYNT*. Bachelor’s thesis, Masaryk University, Brno, Czech Republic, 2007.
- [Haj98] Jan Hajič. Building a syntactically annotated corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*, pages 106–132, Prague, 1998. Karolinum.
- [Hau01] Roland Hausser. *Foundations of Computational Linguistics*. Springer-Verlag, Berlin Heidelberg New York, 2nd edition, 2001.
- [HH06] Dana Hlaváčková and Aleš Horák. Verbalex – new comprehensive lexicon of verb valencies for czech. In *Computer Treat-*

- ment of Slavic and East European Languages*, pages 107–115, Bratislava, Slovakia, 2006.
- [HHK06] Dana Hlaváčková, Aleš Horák, and Vladimír Kadlec. Exploitation of the verbalex verb valency lexicon in the syntactic analysis of czech. In *Proceedings of Text, Speech and Dialogue 2006*, pages 79–85, Brno, 2006. Springer-Verlag.
- [Hor07] Aleš Horák. *Introduction to Computational Linguistics (lecture notes)*. Masaryk University, Brno, Czech Republic, 2007.
- [HSK02] Aleš Horák, Pavel Smrž, and Vladimír Kadlec. Enhancing best analysis selection and parser comparison. In *Proceedings of 5th International Conference on Text, Speech, and Dialogue (TSD 2002)*, pages 461–466, Berlin, 2002.
- [JHK09] Miloš Jakubíček, Aleš Horák, and Vojtěch Kovář. Mining phrases from syntactic analysis. In *Lecture Notes in Artificial Intelligence, Proceedings of Text, Speech and Dialogue 2009*, pages 124–130, Plzeň, Czech Republic, 2009. Springer-Verlag.
- [JKH09] Miloš Jakubíček, Vojtěch Kovář, and Aleš Horák. Measuring coverage of a valency lexicon using full syntactic analysis. In *RASLAN 2009 : Recent Advances in Slavonic Natural Language Processing*, pages 75–79, Brno, 2009.
- [JS97] A.K. Joshi and Y. Schabes. Tree-adjoining grammars, *Handbook of formal languages*, vol. 3: beyond words, 1997.
- [Kad07] Vladimír Kadlec. *Syntactic analysis of natural languages based on context-free grammar backbone*. PhD thesis, Fakulta informatiky, Masarykova univerzita, Brno, září 2007.
- [KH05] Vladimír Kadlec and Aleš Horák. New meta-grammar constructs in czech language parser synt. In *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005.
- [KH07] Vojtěch Kovář and Aleš Horák. Reducing the number of resulting parsing trees for the czech language using the beautified chart method. In *Proceedings of 3rd Language and Technology Conference. Poznań*, pages 433–437. Wydawnictwo Poznańskie, 2007.

- [KHJ09] Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. Syntactic analysis as pattern matching: The SET parsing system. In *Proceedings of the 4th Language & Technology Conference*, pages 100–104, Poznaň, Poland, 2009.
- [KJ08] Vojtěch Kovář and Miloš Jakubíček. Test suite for the czech parser synt. In *Proceedings of Recent Advances in Slavonic Natural Language Processing 2008*, pages 63–70, Brno, 2008.
- [KJ09] Vojtěch Kovář and Miloš Jakubíček. Prague dependency treebank annotation errors: A preliminary analysis. In *RASLAN 2009 : Recent Advances in Slavonic Natural Language Processing*, pages 101–108, Brno, 2009.
- [MSS⁺09] Yusuke Miyao, Kenji Sagae, Rune Saetre, Takuya Matsuzaki, and Jun’ichi Tsujii. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics*, 25(3):394–400, 2009.
- [Ně06] Ivana Němcová. Exploitation of the verbalex verb valency lexicon in the syntactic analysis of czech. Master’s thesis, Masaryk University, Brno, Czech Republic, 2006.
- [PH08] Karel Pala and Aleš Horák. Can complex valency frames be universal? In *RASLAN 2008*, pages 41–48, Brno, 2008. Masarykova Univerzita.
- [PRS97] Karel Pala, Pavel Rychlý, and Pavel Smrž. Desam — annotated corpus for czech. In *Proceedings of SOFSEM’97*, pages 523–530. Springer-Verlag, 1997.
- [PS94] C.J. Pollard and I.A. Sag. *Head-driven phrase structure grammar*. University of Chicago Press, 1994.
- [PS04] Karel Pala and Pavel Smrž. Building Czech wordnet. In *Romanian Journal of Information Science and Technology*, pages 79–88. Romanian Academy, 2004.
- [Sam00] Geoffrey Sampson. A Proposal for Improving the Measurement of Parse Accuracy. *International Journal of Corpus Linguistics*, 5(01):53–68, 2000.

- [SB03] Geoffrey Sampson and Anna Babarczy. A Test of the Leaf-Ancestor Metric for Parse Accuracy. *Natural Language Engineering*, 9(04):365–380, 2003.
- [Ste00] M. Steedman. *The syntactic process*. MIT Press, 2000.
- [Win10] Shuly Wintner. Special event. In *Computational Linguistics and Intelligent Text Processing: 11th International Conference, CICLing 2010*, Iasi, Romania, 2010.