Towards Automatic Extraction of Verb Frames

Ondřej Bojar

Abstract

This article explores the possibilities of automatic extraction of both surface and valency frames of Czech verbs. First, it is clearly documented that the data from Prague Dependency Treebank is not sufficient for collecting enough examples of verb frames to build a large scale lexicon. As a solution, an approach to pick nice examples of sentences from any texts is suggested and thoroughly described. A new scripting language to simplify the selection of sentences based on linguistic criteria was implemented and its main concepts are presented here, too. Also the problems of extracting surface and valency frames from the collected data are addressed and illustrated on real corpus data.

1 Motivation

At the current stage of the development, the accuracy of syntactic analysers of natural languages (in particular Czech) is limited due to the lack of large and precise lexicons of syntactic behaviour of individual words (verb valency frames are the most important example). Although some electronically available lexicons of Czech verbs exist (BRIEF for example), the information provided was collected by hand and suffers an important problem: The theoretical background for the lexicons was purely linguistic and its explanation for computational issues is rather complicated. The lexicons are therefore updated and new ones are built reflecting a theory more precise in the computational field as well. Unfortunately building such lexicons by hand is rather a time-consuming task, so that any kind of automatic preprocessing would help.

This article describes the overall scenario and the main problems of extracting verb frames from corpora. The presented work was done within the framework of Functional Generative Description (FGD[1]). In section 2 I summarize the basic notions of FGD relevant to verb frames extraction. In the following sections 3, 4 and 5 the extraction process and its problems are explained.

2 Basic Notions

This section summarizes FGD notions relevant for extracting verb frames. For a full description of FGD please refer to the books cited.

Levels of language description. FGD defines several levels (layers) of description of sentences of natural languages. On the *morphological* (morphemic) level of annotation, a sentence is represented as a list of word forms equipped with their morphological information. On the *analytic* ("surface syntactic") level of description, a sentence is represented by a dependency tree whose nodes correspond one to one to the word forms in the sentence. On the *tectogrammatical* ("deep syntactic") level of description, also a dependency tree is used but the nodes do not longer correspond to the input word forms. Only the autosemantic words (in their basic form) are represented by nodes in the tree. Also, nodes for elided participants of the sentence are added.

[1] (Sgall, 1967; Panevová, 1980; Sgall, Hajičová, and Panevová, 1986; Panevová, Hajičová, and Sgall, 2001)

Participants and free modifiers. The FGD defines the distinction between *participants* (actants, inner participants, arguments) and *free modifiers* (adjuncts) of a verb strictly on the tectogrammatical level (and not on the analytic level):

- A participant is characteristic of a verb whereas a free adjunct can modify nearly any verb.
- A participant cannot modify a verb twice within a sentence whereas a free adjunct can be used repeatedly.

The set of participants is closed in FGD. The participants are: ACT (actor), PAT (patient), ADDR (addressee), ORIG (origin) and EFF (effect).

For automatic distinction of participants and free modifiers, it is enough to find out the type of the modifier and check if it is one of the participants. The section 5 discusses some problems of this goal.

Obligatory and optional modifiers. The distinction between obligatory and optional modifiers is again defined on the tectogrammatical level only. To summarize the *dialogue test* by Panevová (1980), the modifier is obligatory if its value must be known to the speaker, although the speaker might decide not to express it explicitly on the surface level. This test cannot be performed by a machine.

Observed frame. The list of word forms depending on a verb in an analytic tree of a sentence is called an *observed frame* of the verb. The section 3 analyzes the data available to observe verb frames and describes an approach to collect as many of them as possible.

Surface frame. The set of *typical* word forms modifying a verb is called a *surface frame* or a subcategorization frame. Usually the word forms in a surface frame are described only by their morphological characteristics (case, preposition etc.) For procedures of automatic *surface* syntactic analysis, the lexicon of surface frames would be sufficient. The section 4 suggests a way to convert the set of observed frames to the surface frames of a verb.

Valency frame. The FGD defines *valency frame* of a verb at the tectogrammatical level only. It is the set of *participants* and *obligatory free modifiers* of the verb. (Note that a verb can have different valency frames, they should correspond to the different "meanings" of the verb.) The lexicon of valency frames is needed for all systems aiming at deeper syntactic analysis of input sentences. The section 5 describes main problems with automatic extraction of verb valency frames.

3 Picking Nice Examples

In this section, I describe a novel approach to gaining enough source data for observing verb frames. The main idea is to use not just the syntactically annotated sentences available in a treebank, but to use any texts and select sentences suitable to observe verb frames.

The necessity of such an approach is clearly documented in the section 3.1. The section 3.2 then describes the exact rules used to select the "nice" sentences. The sections 3.3 and 3.4 summarize the results of the described preselection.

A new scripting language AX was designed and implemented to express the rules, a brief description of AX is given separately, in the section 6.

3.1 The Necessity of Picking Nice Examples

Observing verb frames in treebanks (such as the Prague Dependency Treebank, PDT[2], Böhmová et al. (2001)) is a simple task. The word forms depending on a verb are explicitly marked in the tree structure.

[2] `http://ufal.mff.cuni.cz/pdt/`

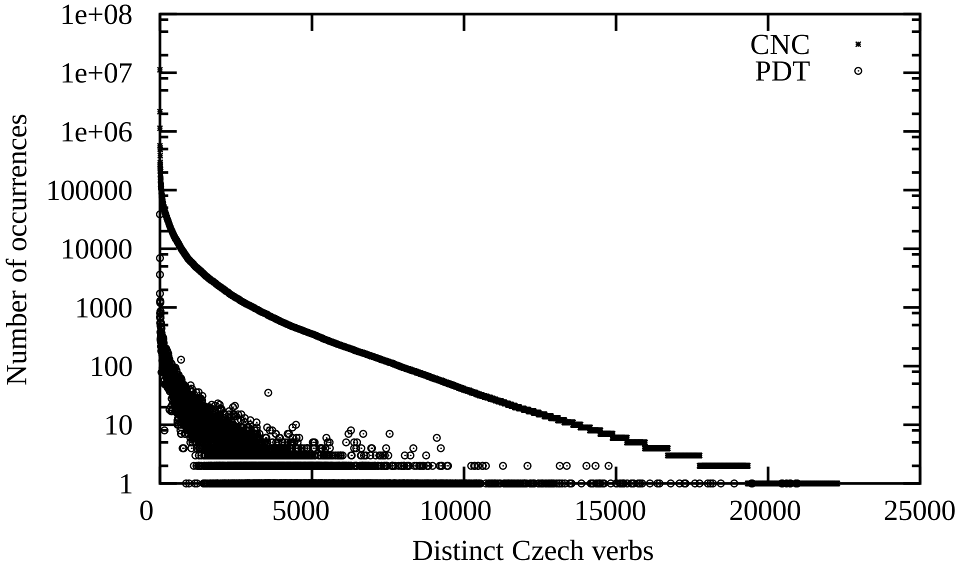


Figure 1: Number of occurrences of verb in PDT and CNC.

The main limitation of the PDT with respect to extracting verb frames is its size. One can try to observe verb frames from bigger corpora (such as the Czech National Corpus, CNC[3]), but there is not the tree structure available.

The Figure 1 compares the basic statistics of Czech verbs observed in the PDT and in the CNC. There are 22,276 different Czech verbs covered in the CNC (the total number of Czech verbs is difficult to estimate, but is expected to be around 40,000). The PDT covers only 5,407 of these verbs and only for a few hundreds of verbs, the PDT contains more than 50 occurrences per verb.

The Table 3.1 shows a detail of the availability of Czech verbs in both the CNC and the PDT. The verbs were ordered by decreasing frequency in the CNC and this list was divided into groups of roughly equal frequencies (the total number of occurrences of all the verbs in a group should be roughly equal for all the groups). The availability of the verbs and their frequencies in the PDT were also explored. The Table 3.1 is a detailed view of the last group from the Table 3.1.

The first column of the Tables is the number of the group. The second column shows the number of distinct verbs that fall into the group. The third column of the Tables show the minimum, average and maximum number of occurrences of the verbs of the corresponding group in the CNC. For instance the most frequent verb *být*$_{to\ be}$ with more than 11 million occurrences in the CNC has a group for itself. On the other hand, the group 10.10 contains more than 16 thousand different verbs, every of which has less than 208 occurrences in the CNC.

The columns 4 and 5 are devoted to observations of verbs in PDT. The column *PDT-Verbs* shows the number of the verbs of the given group, that were seen PDT at least once too. For example, out of the 1,629 verbs from the group 10.9, only 664 (i.e. 40.8% of 1,629) were seen in PDT. The next collumn shows the minimum, average and maximum number of occurrences of the verbs of the corresponding group. For example, already the verbs in the group 7 occur in the PDT on average 96.0 times and the verbs in the group 8 have in the PDT on average only 48.2 examples.

For native speakers, the last column lists some examples of the verbs in the given group. This is to illustrate that even the verbs with relatively low frequency of occurrence are felt quite “common” and well known to the speaker. Therefore, the aim of building a lexicon of verb frames cannot be restricted to frequently used verbs only.

[3] `http://ucnk.ff.cuni.cz/`

		Occurrences in the CNC	**PDT**		
Group	**Verbs**	(min; ∅; max)	**Verbs**	**Occurrences**	**Examples**
1	1	11253207; 11253207.0; 11253207	1 (100.0%)	38646; 38646.0; 38646	být
2	1	2175254; 2175254.0; 2175254	1 (100.0%)	6967; 6967.0; 6967	mít
3	2	570234; 851099.5; 1131965	2 (100.0%)	1725; 2682.5; 3640	moci, muset
4	21	140362; 243575.3; 522307	21 (100.0%)	270; 672.7; 1300	říci, chtít, jít, dát, uvést
5	53	68535; 92773.1; 126411	53 (100.0%)	79; 285.1; 545	čekat, zůstat, znamenat
6	99	40248; 50606.4; 68004	98 (99.0%)	8; 163.3; 316	představit, věnovat, vyjít
7	164	23011; 30707.7; 40210	163 (99.4%)	18; 96.0; 199	číst, přicházet, končit
8	317	10970; 15861.7; 22982	316 (99.7%)	10; 48.2; 112	dokončit, svědčit, přejít
9	818	3551; 6137.0; 10966	814 (99.5%)	1; 18.2; 129	uznávat, dokazovat, vyvinout
10	20800	0; 241.7; 3548	3942 (19.0%)	0; 3.1; 35	rýsovat, vyčistit, najmout

Table 1: Czech verbs available in the CNC and PDT, grouped by frequency in the CNC.

		Occurr. in the CNC	**PDT**		
Group	**Verbs**	(min, ∅, max)	**Verbs**	**Occurrences**	**Examples**
10.1	153	3050; 3298.9; 3548	149 (97.4%)	1; 9.2; 20	rýsovat, vyčistit, najmout
10.2	177	2628; 2835.6; 3048	174 (98.3%)	1; 7.1; 22	sklízet, napít, probouzet
10.3	211	2172; 2385.6; 2627	206 (97.6%)	1; 6.4; 23	vynášet, usnadňovat, předpovědět
10.4	257	1732; 1950.0; 2172	239 (93.0%)	1; 5.3; 18	navrátit, škrtnout, upřesňovat
10.5	328	1367; 1535.0; 1732	299 (91.2%)	1; 4.5; 21	popíjet, odmlčet, zmapovat
10.6	426	1028; 1177.0; 1366	376 (88.3%)	1; 3.4; 15	rozčílit, tyčit, rezervovat
10.7	586	712; 858.0; 1028	451 (77.0%)	1; 2.5; 35	nashromáždit, čarovat, ochladit
10.8	900	440; 558.5; 712	547 (60.8%)	1; 2.0; 10	mrazit, maturovat, zakopnout
10.9	1629	208; 308.5; 440	664 (40.8%)	1; 1.6; 6	pošpinit, rybařit, odvyknout
10.10	16133	0; 31.2; 208	837 (5.2%)	0; 1.2; 8	ohrnout, vyoperovat, podřimovat

Table 2: Czech verbs with less than 3550 occurrences in the CNC grouped by frequency.

Based on a recent version of valency lexicon for Czech (Žabokrtský et al., 2002; Žabokrtský and Straňáková-Lopatková, 2002) describing c. 2,000 Czech verbs thoroughly, two to four *valency* frames can be expected for a generic verb. However, it should be noted that there is a big difference between frequent and less frequent verbs; for instance the very frequent verb *to have* has already more than 50 different valency frames registered in the lexicon. As described in the following sections, the step from a verb observation to the valency frames of the verb is quite complicated and it is therefore clear that more examples of the verb are required to collect surface or valency frames.

To overcome this lack of syntactically annotated data one could use the parsers available for Czech. However, the accuracy of the current parsers is quite limited. Therefore, I employ the parsers only on sentences that are simple enough to be parsed at a reasonable level of accuracy and that are suitable for extracting verb frames. The rules for such a preselection are described in the next section.

3.2 Rules to Pick Czech Sentences Suitable for Verb Frames Extraction

This section describes a set of rules to select sentences suitable for extraction of verb frames. The rules were implemented in a new scripting language AX (see section 6), 15 filters and 21 rules were needed to code the described filtration.

3.2.1 Complex Punctuation, Numbers etc.

In the first phase I filter out all the sentences containing punctuation marks with difficult syntactic analysis. These include dashes, colons, single parentheses, quotation marks, slashes and other symbols. Similarly, I reject all the sentences containing numbers.

There are two reasons for this restriction. First, the current parsers (statistical or hand written) have in general problems with correct attachment of such symbols. And second, these symbols do not help much when extracting verb frames: the punctuation marks are not part of the frame and the numbers are too ambiguous with respect to the morphological categories (especially case).

3.2.2 Combining Analytical Verb Forms

For the purposes of the following phases, it is necessary to combine all the parts of analytical verb forms. This task can be very complicated in a general case, because in Czech it is possible to put even a subclause between the parts of a single verb:

(1) Včera jsem, omlouvám se, zapomněl na naši schůzku.
Yesterday I_have, I_apologize myself, forgotten about our meeting.
I'm sorry that I forgot about our meeting yesterday.

However, sentences with complicated subclause structure will be rejected in the following phases anyway, so I can simplify the rules and combine the verb parts only if there is no other verb between them.

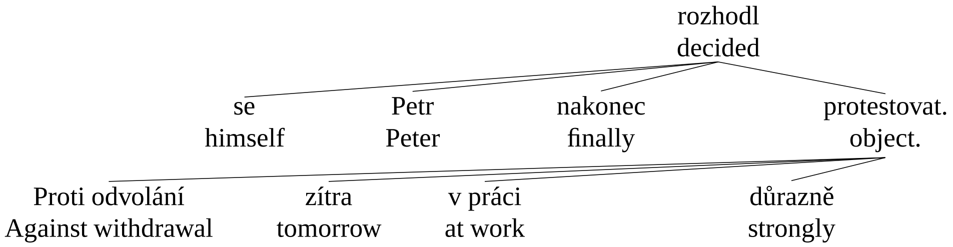
Modal verbs are combined with their autosemantic complements during this phase, too.

3.2.3 Clauses with More Autosemantic Verbs

All the sentences containing two or more autosemantic verbs in a clause are a bad data source when extracting frames of Czech verbs. In Czech, the word order is relatively free and the complements of the verbs can be nearly arbitrarily intermixed.

The example 2 (from Holan et al. (2001), modified) shows the semantically preferred syntax analysis of a sentence with very complex word order. The modifiers *tomorrow*, *at work*, *finally* and *strongly* serve as adjuncts and pure syntactic criteria cannot decide which of the verbs they modify. The modifier *against the withdrawal* is a complement of the verb *to object*, but with a different lexical setting (such as *against expectations*) it could serve as an adjunct of the verb *to decide*. In general, a form of valency information of the verbs in question must be employed. On the other hand, only purely syntactic rules require the complements *himself* and *Peter* to modify the verb *to decide*.

(2)



Peter has finally decided to strongly object against the withdrawal at work tomorrow.

Therefore, I reject all the sentences with two or more verbs (analytical forms have been already combined) where there is no comma or conjunction between them. This phase will not filter out sentences with two verbs in a clause if there is a fake clause delimiter between them (such as a coordinated nounphrase). Later, after combining simple coordination in sentences, this filtering must be repeated.

3.2.4 Coordinated Noun and Prepositional Phrases

Coordination in sentences of natural languages is a very complex phenomenon[4], however current parsers are capable of analysing only very simple forms of coordination. Therefore I perform a partial analysis of the most simple coordination–coordinated noun an preposition phrases–and then (after identifying clauses, see below) reject all the sentences that contain “unexplained” coordination markers such as commas or conjunctions.

The simplified set of rules reduces “prototypical” phrases with coordination step by step:

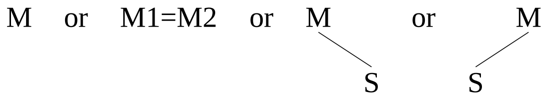
- coordinated adverbs into a single representant,
- an adverb with the following adjective,
- coordinated adjectives into a single representant,
- congruent adjectives to the following (congruent) noun,
- coordinated nouns or pronouns into a single representant,
- a preposition to the following noun or pronoun.

Finally, all noun phrases in genitive following a noun phrase are combined with the preceding (governing) noun phrase. The chaining of noun phrases in genitive is a highly unambiguous syntactic construction in Czech, so no serious error is made when combining all elements of the chain together.

3.2.5 Simple Structure of Subclauses

The boundaries of subclauses can be identified now, because the analytical verb forms were already reduced.

The structure of subclauses in a Czech sentence can be quite complex. For example, a comma at the end of a subclause can mark the end either of the last subclause, or of a preceding governing subclause. Therefore it is not possible to tell whether the words after such a comma belong still to the governing subclause or not. For simplicity, I formulate a filter to reject all the sentences that do not match any of the following simple subclause structure (M stands for a main clause, S stands for a subclause):



[4] The coordination was well described for German already by Kunze (1972) and the same description is applicable for Czech too: Let P, Q, R be segments of a correct sentence. Let P, Q', R be segments of a different correct sentence (i.e. the sentences differ in Q and Q' only; P or R can be optionally blank). Then also the sentence of form P, Q *and* Q', R is a correct sentence of the language. Here is an example:

(3) a. Peter has found a book for Martin.

b. Peter has bought a book for Martin.

c. Peter has found and bought a book for Martin.

The correct understanding of the sentence claims that the book was both found and bought (by Peter, for Martin). In pure dependency syntactic structure, the word *book* should therefore depend on both of the verbs at the same time, but this would violate even the tree property of the structure.

Both in the main clauses and in the subclauses, exactly one verb is required (see 3.2.3) and the boundary of the clauses must be explicitly marked by a comma and/or a conjunction according to the grammatical rules for Czech.

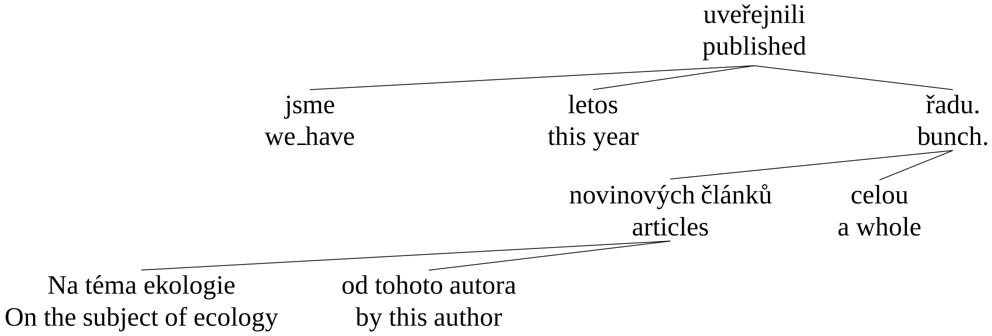
3.2.6 Syntactic Ambiguity of Prepositional Phrases

Straňáková-Lopatková (2001) thoroughly analyses the syntactic ambiguity of prepositional phrases in Czech and tries to formulate clear linguistic criteria applicable in systems of automatic syntactic analysis. She defines "basic disambiguous" and "basic suspicious" word order patterns (WOPs). Thanks to the preprocessing of sentences in the preceding phases of filtration (combining analytical verb forms and grouping noun phrases), I can simply check and reject all the sentences with any of the suspicious WOPs that might influence the observation of verb frames. The following patterns are checked:

- **VNPg** – a verb followed by a noun phrase (possibly having a preposition) and a prepositional phrase. The prepositional phrase can depend on either, the verb or the noun phrase.
- **NPgV** – a noun phrase (possibly having a preposition), a prepositional phrase and a verb. The same type of ambiguity.
- **VPgA** and **PgAV** – a verb followed by a prepositional phrase and an adjective or a prepositional phrase, an adjective and a verb. The prepositional phrase can depend on the adjective or the verb.[5]

The example 4 by Karel Oliva cited by Straňáková-Lopatková (2001) warns that even sentences with no suspicious WOPs and with a single verb only can be very complicated[6]. One of the reasons is that the preferred syntactic analysis is based on valency and semantic criteria too, contrary to the example 2 the valency of nouns must be employed to solve the ambiguity:

(4)



This year we have published a whole bunch of articles by this author on the subject of ecology.

3.3 The Availability of "Very Simple Sentences" in Corpora

The results of the filtering process described in the previous section are shown in the Table 3.3. Out of the first 144 839 sentences of the CNC, 15 to 20% sentences (depending on whether the suspicious WOPs are checked or not) pass all the filters. For simplicity, I call the selected sentences "very simple".

The Table 3.3 summarizes the "strength" of distinct filtration phases. In the first column, the number of sentences filtered out in the given phase is shown. The total number of sentences rejected up to this

[5] Straňáková-Lopatková (2001) warns that this WOP should never occur at any stage of the reduction analysis of the sentence. For efficiency issues, I run the filtration process of AX in fully deterministic mode, although AX allows for nondeterministic analysis as well. Therefore, the noun phrases are already fully combined when applying the WOP filter. I expect the error caused by this simplification is not too big, but a more serious analysis would be needed for a confirmation.

[6] See (Holan et al., 1998; Holan et al., 2001) for a definition of an exact measure of nonprojectivity of dependency trees.

point of filtration is given in the second column. To stress the consecutiveness of the phases, the names of the filters are prefixed by an ascending sequence of letters.

Sentences from the CNC	**Number of sentences**	
	In each phase	**Cumulative**
Rejected-A a dash was found	18 156 (12.5 %)	18 156 (12.5 %)
Rejected-B a single quote	33 (0.0 %)	18 189 (12.6 %)
Rejected-C a slash	1 369 (0.9 %)	19 558 (13.5 %)
Rejected-D a colon in the middle	1 030 (0.7 %)	20 588 (14.2 %)
Rejected-F a semicolon in the middle	947 (0.7 %)	21 535 (14.9 %)
Rejected-G numbers	22 458 (15.5 %)	43 993 (30.4 %)
Rejected-H unmatched parenthesis	548 (0.4 %)	44 541 (30.8 %)
Rejected-J too many verbs	3 594 (2.5 %)	48 135 (33.2 %)
Rejected-K too complex subclauses	66 098 (45.6 %)	114 233 (78.9 %)
Rejected-L suspicion of **VNPg**	5 661 (3.9 %)	119 894 (82.8 %)
Rejected-M suspicion of **NPgV**	1 860 (1.3 %)	121 754 (84.1 %)
Rejected-N suspicion of **PgAV**	218 (0.2 %)	121 972 (84.2 %)
Rejected-O suspicion of **VPgA**	664 (0.5 %)	122 636 (84.7 %)
Selected by the script as a whole	22 203 (15.3 %)	144 839 (100.0 %)
Total number of sentences	144 839 (100.0 %)	144 839 (100.0 %)

Table 3: "Very simple sentences" in the CNC.

3.4 The Utility of Sentence Preselection

To evaluate the utility of the described preselection, the same selection was performed on the evaluation part of the Prague Dependency Treebank. Three parsers available for Czech (Collins et al. (1999), Zeman (1997, 2002) and a parser by Zdeněk Žabokrtský (unpublished)) were then used on all the sentences, and separately on the selected "very simple sentences". The accuracy was measured by three distinct criteria and results of the experiment are shown in Table 4.

		Statistical		**Hand-made**
Observed Frames Correct	**Verbs**	**Collins**	**Zeman**	**Žabokrtský**
All Sentences	16 329	55.32%	33.11%	39.5%
Very simple sentences	2 472	61.37%	41.87%	44.8%
… and no suspicious WOP	1 546	64.68%	47.02%	53.8%
Correct Dependencies	**Words**	**Collins**	**Zeman**	**Žabokrtský**
All Sentences	126 030	82.51%	69.15%	73.8%
Very simple sentences	20 028	87.70%	79.40%	82.3%
… and no suspicious WOP	11 030	87.89%	79.31%	83.6%
Sentences without a Mistake	**Sentences**	**Collins**	**Zeman**	**Žabokrtský**
All Sentences	7 319	30.95%	15.00%	18.4%
Very simple sentences	1 786	47.14%	29.00%	31.6%
… and no suspicious WOP	1 113	52.83%	34.41%	41.5%

Table 4: Accuracy of three parsers of Czech on the evaluation part of PDT compared to accuracy achieved on the selected "very simple sentences" only.

Bearing in mind the task of extracting valency frames, the most important criterion is the number of correctly observed frames, that is the number of verbs, that have correctly assigned all the daughters.

The results show that the best parser available for Czech, the Collins parser, is able to correctly assign daughters to 55% of occurrences of verbs. When used on very simple sentences only, this measure increases by 10% to 65%. Also quite interesting is the improvement of the traditional accuracy measure, namely the number of correctly assigned dependencies. If used on very simple sentences, the parsers achieve an accuracy of 5 to 10 percent of correctly assigned dependencies better, up to 88% for Collins.

The last criterion, which could be important for the task of extraction of different lexico-syntactic type of information, is the number of sentences parsed without a mistake. If only very simple sentences are analyzed, the parsers by Zeman and Žabokrtský achieve more than double the accuracy by this measure and the parser by Collins analyses more than 50% of sentences without a mistake, instead of 30% if run on all sentences.

Some examples of “very simple sentences” are listed in the sample output of AX in the Figure 5.

4 From Observed fo Surface Frames

This section is focussed on the step of finding out surface (subcategorization) frames given the set of observed frames of a verb (see section 2 for the definitions). This is quite a complicated task, because verbs often occur modified by free modifiers (adjuncts) and the required modifiers are often missing in the observed frame. Several statistical approaches are described in literature, for instance (Brent, 1991; Manning, 1993; Sarkar and Zeman, 2000; Briscoe and Carroll, 1997), but they often presume that the correct surface frame is actually present among the observed frames. Sarkar and Zeman (2000) point out that this assumption is not always valid, free modifiers of different forms are often present in all the observed frames in Czech.

Furthermore, Korhonen, Gorrell, and McCarthy (2000) evaluate several such methods and document the accuracy of 50%. (Surface frames extraced by an algorithm are compared to the surface frames extracted by a human annotator.) The most alarming fact is that simply selecting the most frequent frames up to a threshold chosen by the annotator achieves c. 75% of accuracy.

Therefore, I prefer to employ linguistic filtration and human-controlled analysis of the observed frames.

4.1 Simple Linguistic Filtration

Some of the modifiers observed under a verb can be ignored as linguistically irrelevant for the surface frame of the verb. I immediately remove modifiers of the following kinds, because they can in general modify any verb and should not be listed in lexicons:

- rhematizers (the list provided by Hajičová, Panevová, and Sgall (1999))
- particles
- adverbs expressing time (the list is maintained for annotators of PDT)
- causal subclauses (for first approximation, the subclauses starting with conjunctions *protože*, *proč*, *když*, *kdyby*, *pokud* and *aniž*)

Furthermore, time modifiers are identified by searching for a limited set of lexical items. This algorithm was suggested by Zdeněk Žabokrtský and evaluated by Bojar (2002). Bojar documents that on the basis of its topmost lemma it is possible to decide whether a modifier is a time expression or not. This algorithm decides correctly for 96% of modifiers (regardless the specific type of time expressions).

The utility of the described filtration is clearly illustrated by an example of the verb *dát*$_{to\ give}$. Among 232 observed frames, 158 different frames were found (containing together 97 distinct modifiers). After the filtration of modifiers, only 127 frames remained different and the number of distinct modifiers was reduced from 97 to 65.

The zero and first levels: (The listing is complete.)

```
- (10 superframes with 231+1=232 observations)
  - (5 superframes with 124+2=126 observations) se
  - (4 superframes with 122+2=124 observations) infin
  - (11 superframes with 111+1=112 observations) #1
  - (16 superframes with 59+6=65 observations) #4
  - (9 superframes with 33+2=35 observations) #3
  - (5 superframes with 5+3=8 observations) si
  - (3 superframes with 7+1=8 observations) do-1#2
  - (4 superframes with 5+1=6 observations) na-1#4
  - (0 superframes with 0+1=1 observations) jasně_ˆ(*1ý)#Dg najevo#Db že
  - (0 superframes with 0+1=1 observations) 2x(time_modifier)
```

Some deeper levels of the hierarchy:

```
- (10 superframes with 231+1=232 observations)
  - (5 superframes with 124+2=126 observations) se
    - (17 superframes with 92+27=119 observations) infin se
    - (1 superframes with 2+1=3 observations) #7 se
    - (1 superframes with 1+1=2 observations) do-1#2 se
    - (0 superframes with 0+1=1 observations) #1 na-1#4 se
    - (0 superframes with 0+1=1 observations) #1 pak#Db se v-1#4
  - (4 superframes with 122+2=124 observations) infin
    - (17 superframes with 92+27=119 observations) infin se
    - (1 superframes with 1+1=2 observations) #1 time_modifier infin
    - (0 superframes with 0+1=1 observations) #1 #4 aby infin podle-2#2
    - (0 superframes with 0+1=1 observations) ADJ#- infin
  - (11 superframes with 111+1=112 observations) #1
  - (16 superframes with 59+6=65 observations) #4
  - (9 superframes with 33+2=35 observations) #3
  - (5 superframes with 5+3=8 observations) si
    - (0 superframes with 0+1=1 observations) #4 během#2 si
    - (0 superframes with 0+1=1 observations) #1 #4 time_modifier si
    - (0 superframes with 0+1=1 observations) #4 k-1#3 si
    - (0 superframes with 0+1=1 observations) #4 kromě#2 na-1#6 si z-1#2
    - (0 superframes with 0+1=1 observations) #4 numeral#- si spolu#Db
  - (3 superframes with 7+1=8 observations) do-1#2
  ...
```

Figure 2: The hierarchy of observed frames of the verb $dát_{to\ give}$. The character # is used to mark the morphological case of the modifier, prepositions are listed before this mark. The topmost blank frame was directly observed only once (the `+1` in the list), but all its "superframes", i.e. the frames containing the frame as well, were altogether observed 231 times. 10 different types of frames are the closest neighbours of the blank frame and are listed in the first level of the hierarchy.

4.2 Hierarchical Browsing of Observed Frames

To simplify browsing of the obtained list of (filtered) observed frames, the frames are ordered into a hierarchy. The hierarchy starts with a blank frame (frame containing no modifiers at all). Given a frame in the hierarchy, all the frames containing the same modifiers and some extra modifiers are called *superframes* of the frame.

See Figure 2 for an example of the frames observed for the verb *dát*$_{to\ give}$. Browsing such a hierarchy in a text editor could also help a human annotator of verb frames.

4.3 An Automatic Examination of the Frame Hierarchy

In order to identify surface frames automatically, I implemented a simple automatic procedure to browse the hierarchy of observed frames:

- Every frame that was observed at least n times (including all its superframes) is replaced by its immediate superframes and the superframes are examined again.
- Only the more frequent superframes are studied, that is the superframes whose number of observations (incl. all the children in the hierarchy) reaches at least p percent of the total observations of the governing frame (i.e. the immediate subframe).
- Expand the hierarchy recursively in the described manner, until all the frames were either observed less than n times or all their sons (immediate superframes) are too equally distributed and none of them exceeds p percent of the occurrences of the immediate subframe.

Finally, the frames with which the expansion process has stopped are listed as the estimated surface frames. (Some of the frames could be listed more than once, as several paths in the hierarchy can lead to a single frame.)

Sample results of the algorithm employed on the hiearchy of frames observed for the verb *dát*$_{to\ give}$ are shown for the setups of $n = 20$ and $p = 10$ and 5 in the Table 4.3. The number of observations of the frame itself is listed in the column *Own occurrences*. The number of observations of superframes of the frame is listed in the column *Occurrences of frames with more modifiers* (these are expected to be the occurrences of the frame with some extra adjuncts) and the number of observations of subframes is in the column *Occurrences of frames with less modifiers* (these approximate the occurrences of the frame with some of the members elided). No estimate is provided for occurrences of the frame with both, extra adjuncts and some modifiers missing. The sum of the first three columns is given in the column *The estimated occurrences of the frame* and the percentage of this estimate within the total number of occurrences of the verb is listed in the last column. Naturally, the sum of all the percentages does not need to be equal to 100%.

These preliminary results of conversion from observed to surface frames are quite satisfactory if a human annotator is expected to complete the extraction. However, for fully automatic frames extraction, an extensive research and evaluation would be still needed.

5 Towards Valency Frames

As I briefly described in the section 2, the first step necessary to extract valency frames of verbs is to identify the type (the tectogrammatical function, the role) of modifiers. The three following factors must be considered when trying to perform such a decision automatically:

- The surface realization of the modifier. (For dependent nouns, this generally means the case and preposition.)

Frames estimated for $p = 10$	Own occurrences	Occurrences of frames with more modifiers	Occurrences of frames with less modifiers	The estimated occurrences of the frame	The percentage of occurrences of this frame within the total number of occurrences of the verb
#1 #3 #4	8	8	19	35	15,0 %
#1 subclause infin se	5	2	66	73	31,3 %
#1 #4 time_modifier	2	4	14	20	8,6 %
#1 infin podle-2#2 se	5	1	59	65	27,9 %
Frames estimated for $p = 5$					
#1 #3 #4	8	8	19	35	15,0 %
infin se že	5	5	32	42	18,0 %
#1 subclause infin se	5	2	66	73	31,3 %
#1 #4 time_modifier	2	4	14	20	8,6 %
#1 najevo#Db že	3	3	2	8	3,4 %
#1 infin podle-2#2 se	5	1	59	65	27,9 %
#1 #3 za-1#4	3	0	4	7	3,0 %
#1 infin se v-1#6	3	0	58	61	26,2 %
#1 #4 v-1#6	1	2	15	18	7,7 %
#1 #4 do-1#2	1	1	15	17	7,3 %
#3 #4 subclause	1	1	14	16	6,9 %
#1 #4 #7	1	1	16	18	7,7 %
#1 #4 na-1#4	1	1	14	16	6,9 %
#1 #4 z-1#2	1	1	13	15	6,4 %

Table 5: A sample of automatic exploration of the frame hierarchy observed for the verb *dát*$_{to\ give}$. A detailed explanation is in the section 4.3.

- The lexical value of the verb. (If only active forms of verbs are taken into account; otherwise also the diathesis must be considered.)
- The lexical items in the modifier.

5.1 The Surface Realization of the Modifier

The surface realization of a verb modifier (i.e. the case and preposition for nouns) is a good cue for restricting the set of possible tectogrammatical functions. However, the surface realization itself induces the function in an unambiguous way very rarely.

As the annotation of the PDT on the tectogrammatical level is now in progress (with c. 26,000 sentences ready) and as the valency lexicon of Czech is being developed, preliminary estimations measuring the ambiguity of the functions of modifiers can be made. The results are not unexpected, but anyway quite unsatisfactory from a viewpoint of automatic processing: for example, a nominative governed by an active verb has primary function of ACT (the actor) only in 91% of occurrences. For other roles, the ambiguity is always even worse. This unpleasant estimate could be confirmed from the valency lexicon of Czech: the nominative as the modifier of a verb is marked as ACT only for 97% of c. 2,000 of verbs covered by the lexicon recently. For 2% of the verbs, a nominative plays the role of PAT and exceptionally it is used as other participants or free modifiers as well (EFF, COMPL, MANN and BEN).

5.2 The Lexical Value of the Observed Verb

When trying to extract valency frames of verbs automatically, an important assumption is usually made. One assumes that the lexical value of the analyzed verb is not significant (or the impact is weak enough), i.e. that one can use the same inferencing algorithm for all the verbs.

However, it is not too difficult to find examples of sentences in which the choice of the verb is the only difference but the modifiers of the verb should be treated differently:

(5) a. Posuň ten obrázek $<$ke konci stránky$>_{\mathtt{DIR3}}$.

b. Nechej ten obrázek $<$ke konci stránky$>_{\mathtt{LOC}}$.[7]

Move/keep the picture $<$to the end of the page$>_{\mathtt{DIR3/LOC}}$.

The Table 5 stresses my argument that the lexical value of the analyzed verb cannot be blindly neglected when extracting valency frames. The 26,000 of sentences of PDT annotated at the tectogrammatical level were scanned for verbs and their modifiers. In the first column, the most frequent forms of modifiers are listed (all the listed forms were observed at least in 500 different occurrences). The second column (*Observed with verbs*) shows the number of distinct verbs with a modifier of the given form was observed. This total number of verbs is then divided into disjoint groups: the group of verbs with all the modifiers of the given form were observed in the role of ACT, PAT etc; the group of verbs with the modifiers of the given form were seen in the role of more different participants (*Mixture of partic.*), the group of verbs with all the modifiers of the given form were seen in a role of a free modifier regardless the type of the free modifier (*Free modifiers only*) and finally the group of verbs with the modifiers of the given form were used as both, participants and free modifiers (*Partic. too*).

For example, the nominative *#1* was observed with 2,487 different verbs. With 79.3% of these verbs, it always played the role of ACT, but with 2.5% of these verbs, it always played the role of PAT (these include examples of verbs in reflexive passive form, which is difficult to distinguish automatically from active verb forms). Similarly, the dative *#3* served as an ADDR for 45.2% of verbs (such as *znepříjemnit*, *nařídit*), but for 18.1% of verbs it always played the role of PAT[8] (*podlehnout*, *vyhovovat*, *sloužit*). The most difficult form to analyse is an infinitive: for 38.1% of verbs it is a clear PAT (*považovat*, *cítit*, *navrhovat*), but for 30.4% of verbs it is a clear free modifier regardless its type (*zatelefonovat*, *preferovat*).

5.2.1 The Lexical Items of the Modifier

It is difficult to define an exact algorithm inferring the function of a modifier and utilising the observed lexical items of the modifier. A good inspiration comes from the BRIEF lexicon in which animate and inanimate modifiers are distinguished ("beings" vs. "objects"), however, the morphological attributes do not always correspond to the real property of animacy (e.g. for institutions etc.).

[7] Although the example might sound a bit strange to a native speaker, the expression *ke konci stránky* marked as LOC is explicitly listed in Hajičová, Panevová, and Sgall (1999).

[8] This is probably related with the issue of "shifting" of participants, a part of FGD: if the verb has a single participant, then it should be marked as ACT. If the verb has two participants, they should be marked as ACT and PAT, regardless the form of the participants. If there are two verbs with similar semantic properties, one having three participants (incl. one in the dative form) and the other has only two, then the second participant of the second verb should be marked as PAT, even if it is semantically derived from the ADDR-dative participant of the first verb.

	Observed		**Always seen as participant**				**Mixture**	**Free modifiers**	
Form	**with verbs**	ACT	PAT	ADDR	ORIG	EFF	**of partic.**	Free only	Partic. too
#1	2 487 (81,9 %)	79,3 %	2,5 %	-	-	0,1 %	10,9 %	0,6 %	6,7 %
#4	1 819 (59,9 %)	1,4 %	78,9 %	2,2 %	-	0,4 %	4,5 %	3,0 %	9,6 %
subclause	1 168 (38,5 %)	1,0 %	9,8 %	-	0,1 %	1,4 %	1,1 %	64,6 %	22,0 %
v#6	988 (32,5 %)	-	0,9 %	-	-	-	-	96,4 %	2,7 %
se	940 (31,0 %)	-	4,7 %	0,4 %	-	-	-	90,5 %	4,4 %
že	610 (20,1 %)	-	-	-	-	-	-	100,0 %	-
#7	585 (19,3 %)	0,3 %	7,2 %	-	-	2,1 %	0,2 %	83,4 %	6,8 %
#2	545 (18,0 %)	19,1 %	40,9 %	1,8 %	-	0,6 %	5,7 %	17,6 %	14,3 %
#3	476 (15,7 %)	1,7 %	18,1 %	45,2 %	0,2 %	-	6,7 %	22,9 %	5,3 %
na#6	459 (15,1 %)	-	3,5 %	0,4 %	0,7 %	-	0,2 %	91,5 %	3,7 %
na#4	402 (13,2 %)	0,7 %	21,9 %	1,7 %	-	2,7 %	1,0 %	62,9 %	9,0 %
i-1	374 (12,3 %)	-	-	-	-	-	-	100,0 %	-
infin	365 (12,0 %)	3,6 %	38,1 %	-	-	6,6 %	5,2 %	30,4 %	16,2 %
do#2	363 (12,0 %)	-	1,9 %	-	-	1,9 %	-	92,3 %	3,9 %
#X	351 (11,6 %)	61,8 %	14,5 %	4,6 %	-	0,6 %	7,1 %	7,1 %	4,3 %
s#7	348 (11,5 %)	-	19,5 %	6,6 %	-	2,3 %	2,9 %	62,1 %	6,6 %
z#2	317 (10,4 %)	-	6,0 %	-	6,3 %	-	0,3 %	82,0 %	5,4 %
však	292 (9,6 %)	-	-	-	-	-	-	100,0 %	-
po#6	284 (9,4 %)	-	2,1 %	-	0,7 %	-	-	94,0 %	3,2 %
podle#2	282 (9,3 %)	-	-	-	-	-	-	100,0 %	-

Table 6: The impact of the lexical value of verbs on the function of different forms of modifications. Only verbs in active diathesis were analyzed.

Further research would be also needed in order to employ for example the classification of lexical items as provided in the Eurowordnet. Using such a classification would definitely help to reduce the observed types of items and this could simplify the task of attributing a function to an observed modifier.

As briefly described in the section 4.1, the observed lexical items can be successfully used to distinguish time modifiers.

6 AX – A System to Pick Sentences Matching Linguistic Criteria

The system AX was developed to simplify the task of selecting feasible examples of sentences for extracting a specific lexico-syntactic information. The sentences can be easily selected on linguistically based criteria. Partial syntactic analysis of sentences is possible, in order to be able to answer more complex linguistic questions about the sentence.

6.1 The Architecture of AX

After startup, the system AX loads a script with filters and rules and then expects sentences augmented with their morphological annotation (in the format of the PDT[9]) on the standard input. The input sentences may be morphologically disambiguated or not. For every input sentence, the system runs the script and checks if the sentence passed all the filters or has been rejected. For sentences that pass (let's call them the "selected sentences"), the ID and the output of the final phase is printed out. This output is for some purposes already suitable for collecting the lexico-syntactic information so that no other parser to process the sentences is needed.

In the following, I describe the overall running scheme of AX. The input sentence is internally stored as a sequence of feature structures that correspond one to one to input word forms. (See section 6.2 for details.) The input sentence is then processed through a pipe of consecutive blocks (phases) of operation. Each of the blocks is either a filter, or a set of rules. The input for each block is a set of

[9] See `http://shadow.ms.mff.cuni.cz/pdt/Corpora/PDT_1.0/Doc/morph.html`

sequences of feature structures (let's call it the set of "input readings" of the sentence). If the block is a filter, it checks all the input readings and possibly rejects some of them. If the block is a set of rules, it updates every input reading with all applicable rules and returns a larger set of new readings (it "generates" new readings). Consecutive blocks are connected, so that the output set of readings from the former block is used as the input set of readings for the latter block. The first of the blocks receives as input the input sentence, the output from the last block is printed out. The order and type of the blocks is up to the author of the script. A sample flow of readings is demonstrated in the Figure 3, a sample output is in the Figure 5.

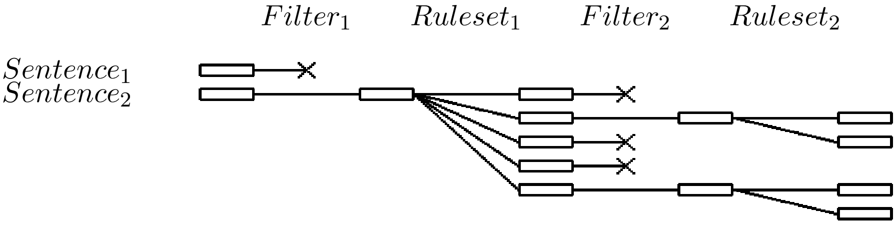


Figure 3: Progress of sentences through an AX script. The first input sentence was rejected by the first filter. The second sentence passed the filter and several readings were obtained by the rules in ruleset 1. Some of the readings were then rejected by filter 2 and some passed. Altogether four different readings were then produced by the last ruleset.

6.2 Feature Structures with Variants

Feature structures (also called attribute-value matrices) are data structures often used for describing linguistic phenomena. A detailed characteristics of typed feature structures is given by Penn (2000), for the purposes of this work, untyped feature structures are sufficient. On the other hand, I augment the definition by allowing alternatives (variants) in values. A *variant feature structure* is one of:

1. A simple value (such as sg to represent singular or int(312) to represent a number),
2. A list of tuples attribute - value, where the attribute is a string name (unique within the list) and the value is a variant feature structure. The order of the tuples in the list is not significant.
3. A set of possible variants, where every member of the set is a variant feature structure.

The basic operation with two variant feature structures is *unification*. The output of the unification is a feature structure that holds information from both the input structures.[10] Unification fails if both the features contain an attribute of the same name but a nonunifying value. For instance:

$$\begin{bmatrix}\text{name} & \text{Kamil}\\ \text{surname} & \{\text{Horak, Klement}\}\end{bmatrix} \text{ and } \begin{bmatrix}\text{surname} & \text{Horak}\\ \text{age} & \text{int(32)}\end{bmatrix} \begin{array}{c}\text{unify and}\\ \text{the result is}\end{array} \begin{bmatrix}\text{name} & \text{Kamil}\\ \text{surname} & \text{Horak}\\ \text{age} & \text{int(32)}\end{bmatrix}$$

$$\begin{bmatrix}\text{name} & \text{Kamil}\\ \text{surname} & \text{Horak}\end{bmatrix} \text{ and } \begin{bmatrix}\text{name} & \text{Josef}\\ \text{surname} & \text{Horak}\\ \text{age} & \text{int(32)}\end{bmatrix} \text{ do not unify.}$$

For every input word in a sentence, the morphological analysis gives all possible lemmas and morphological attributes of the given word form. This ambiguous morphological information can be stored in a single feature structure with variants. See Figure 4 for an example. The whole sentence of word forms can therefore be stored as a list of feature structures of the same length.

[10] If more variants of a value are available, the output will carry out the intersection (more precisely the product of unification of all possible combinations of input variants).

$$\left\{ \begin{bmatrix} \text{cat} & \text{verb} \\ \text{morfcat} & \text{verb(presfut)} \\ \text{lemma} & \text{string(”mít”)} \\ \text{form} & \text{string(”má”)} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{sg} \\ \text{pers} & \text{third} \end{bmatrix} \\ \text{tense} & \text{pres} \\ \text{voice} & \text{active} \\ \text{neg} & \text{pos} \end{bmatrix}, \begin{bmatrix} \text{cat} & \text{pron} \\ \text{morfcat} & \text{pron(poss)} \\ \text{lemma} & \text{string(”můj”)} \\ \text{form} & \text{string(”má”)} \\ \text{agr} & \left\{ \begin{matrix} \begin{bmatrix} \text{case} & \{\text{nom, vok}\} \\ \text{gend} & \text{fem} \\ \text{num} & \text{sg} \\ \text{pers} & \text{first} \end{bmatrix} \\ \begin{bmatrix} \text{case} & \{\text{nom, aku, vok}\} \\ \text{gend} & \text{neut} \\ \text{num} & \text{pl} \\ \text{pers} & \text{first} \end{bmatrix} \end{matrix} \right\} \end{bmatrix} \right\}$$

```
[
  cat - verb, morfcat-verb(presfut), lemma - "mít", form - "má",
  agr - [ num-sg, pers-third ],
  tense - pres, voice - active, neg-pos
|
  cat - pron, morfcat - pron(poss), lemma - "můj", form - "má",
  agr - [ case - nom;vok, gend - fem, num - sg, pers - first
        | case - nom; aku; vok, gend - neut, num - pl, pers - first ]
]
```

Figure 4: This feature structure represents all the possible morphological analyses of the word form *má* which can be a word form of two different lemmas (*mít*$_{\texttt{verb}}$ and *můj*$_{\texttt{pronoun}}$ in different cases, numbers and genders). Below is the same feature structure expressed in the syntax of the scripting language of AX.

Shortcuts can be defined by means of the directive shortcuts or shortcut for feature structures that are used often. In filters and rules, it is then possible to introduce the whole structure only by its shortcut name. The system AX for Czech also supports as an option so called “instant morphological analysis”: instead of precisely describing all the features of a word form, one can simply enter the word form enclosed in backquotes. The word form will be morphologically analyzed when compiling the script.[11] Here is an example of shortcut definition and instant morphological analysis:

```
shortcuts
  noun_like = [cat-noun | cat-pron,morfcat-pron(pers);pron(pers_short)],
  adj_like = [cat-adj | cat-pron,morfcat-pron(poss);pron(poss_refl)]
end

shortcut jsem = ‘jsem‘
```

6.3 Filters

Filters in the language AX are expressed in the form of *regular expressions of feature structures*. The basic differences between common regular expressions (used for instance in many Unix tools) and regular expressions of feature structures used in the language AX are:

- The primitive element of regular expressions is no longer a character, but rather a feature structure. In scripting language AX, this feature structure can be expressed either explicitly or by means of “instant morphological analysis” or by a shortcut name.
- When searching for a subsequence of feature structures that matches a given regular expression, the system checks whether the input structure unifies with the structure in the expression. (Rather than checking the two characters for equality.)

[11] For this feature to operate properly, the system requires a morphological lexicon of Czech.

Details of the syntax are described in Bojar (2002), here I give just a brief example of two different filters:

```
filter reject_more_than_two_verbs:
  .* verb .* verb .*
end

keep "Keep only sentences with exactly one verb
      or those not containing any conjunction":
  !verb* verb !verb* | !conj*
end
```

The keyword filter means: reject the sentence if it (as a whole) matches the given regular expression. The meaning of the keyword keep is: reject the sentence if it doesn't match the given expression.

6.4 Rules

Rules are used to modify the input readings of a sentence and generate new readings. Rules have always this form:

rule <rule name> :
<replacement> —> <input regular expression> ::
<constraints>
end

If there are no constraints needed for the rule and no name is specified, one can use a shorter form for the rule:

rule <replacement> —> <input regular expression> end

The rule is applied as follows:

- The input sequence of feature structures is searched in order to find a subsequence that matches the <input regular expression> and the <constraints>.
- The obtained subsequence of feature structures is replaced with the <replacement>.

By default, the input sequence is searched for all possible subsequences matching the regular expression and constraints, therefore for one input reading the rule can produce several output readings. This nondeterministic approach can be restricted by writing keyword detstart or detend into the arrow in the rule. Then for detstart, the starting point of the rule is fixed once the rule succeeds and similarly for the detend (this restricts the iteration operators (*) of the regular expression). If both, starting and end point should be fixed, one can write the symbol ! in the arrow.

By default, the output of one rule within a ruleset is used as input for another rule in the same ruleset, including the rule itself. All possible combinations of applying rules are attempted and all the possible outcomes are collected to build the final output set of possible readings for this block (phase) of operation. This nondeterministic behaviour can be restricted in several ways: rules can be limited in number of allowed applications, an output from one rule can be included in the final set only if no other rule was able to change it, and others. A detailed description of all the options is out of the scope of this paper.

In order to "compute" the <replacement> from the subsequence found in the input, one can use *variables*. The variables can be used in all parts of rules: from the <input regular expression> they get their initial value. The value is then restricted or updated by the <constraints> and the final value is given to the output in the <replacement>.

All the variables can hold a feature structure. The scope of the variables is limited for one application of one rule, that is all the variables are local for the rule and within one application.

The <constraints> are expressed as an unordered list of requirements on variables values. All the requirements must be fulfilled for the rule to be applicable. The constraints can only require certain feature (sub)structures to unify. By means of these requirements, output variables also get their value. The following example shows a rule to perform a reduction: it combines an adjective and a noun together:

```
rule out_noun ---> adj noun ::
  adj.agr = noun.agr,
  out_noun = noun
end
```

The constraint adj.agr = noun.agr guarantees the congruence of the noun and the adjective in case, gender and number. The constraint out_noun = noun initializes the output variable with the feature structure of the noun *after* it was already restricted in case, number and gender due to the congruence requirement. In this way, the input morphological ambiguity is solved step by step.

The output <replacement> may copy parts of the input subsequence. This allows an easy formulation of rules that combine distant feature structures in the input sentence. The regular expression can then be used to restrict what can stay between the two (or more) feature structures. As a nice example I show the rule that combines two parts of a Czech verb – the auxiliary part (*být*$_{to\ be}$, which can have several forms, such as *jsem*$_{I\ am}$) and the main verb (such as *zalít*, which also can have several forms). The rule also checks the parts for congruence[12]:

```
rule complex_past_tense:
  complex \gap trace
  ---->
     zalil {gap:!{verb,comma,conj}*} jsem
   | jsem {gap:!{verb, comma, conj}*} zalil
  ::
  zalil <- morfcat, voice -> ‘zalil‘,
  zalil = [cat-verb],
  jsem.cat = ‘jsem‘.cat,
  jsem <- morfcat, lemma, neg -> ‘jsem‘,
  jsem.agr = [pers-first;second],
  zalil <- agr.num, agr.gend -> jsem,
  complex = [cat-complexpast],
  complex <- lemma, form, neg, morfcat -> zalil,
  complex <- agr.pers, agr.num, agr.gend, diathesis -> jsem
  trace = [cat-trace, form-"XtraceX"]
end
```

The rule finds such a subsequence of feature structures that begins with the auxiliary verb and ends with the main verb (or vice versa). The gap between the parts of the verb must not contain any other verb, comma or conjunction (introduced by means of shortcuts, see above). The gap gets a label “gap”, so that it can be copied to the output replacement. As the output, the rule produces a single feature structure representing the complex verb followed by a copy of the gap section and an auxiliary trace at the place where the other part of the verb was found. Naturally, the trace can be omitted if not needed by any other rules of filters.

7 Conclusions

In this article I explored the possibilities of extracting verb frames from corpora in a semiautomatic manner. I documented the necessity of using all data sources possible, and not just the data available

[12] The scripting language AX has a shorter form of expressing several unification requirements on two variables at once. The constraint: “usnul <- cat, agr.num, agr.gend -> jsem” is equivalent with these three: “usnul.cat = jsem.cat, usnul.agr.num = jsem.agr.num, usnul.agr.gend = jsem.agr.gend”

```
ID:  cmpr9410:009-p7s1
IN:  Máme zaměstnance , které občas vysíláme na služební cestu .
OUT: mít [lemma-"zaměstnanec",form-"zaměstnance",
          agr-[case-aku,gend-masc,num-pl],prep-"BLANK"]
OUT: vysílat [lemma-"který",form-"které",
              agr-[case-aku,gend-inanim;masc,num-pl]]
             občas
             [lemma-"cesta",form-"cestu",
              agr-[case-aku,gend-fem,num-sg],prep-"na"]
ID:  cmpr9410:013-p5s4
IN:  Přesně k tomu slouží naše rubrika .
OUT: sloužit Přesně
             [lemma-"ten",form-"tomu",
              agr-[case-dat,gend-neut;masc;inanim,num-sg],prep-"k"]
             [lemma-"rubrika",form-"rubrika",
              agr-[case-nom,gend-fem,num-sg,pers-first],prep-"BLANK"]
ID:  cmpr9410:028-p16s2
IN:  Ceny jejich obrazů šplhají do statisíců a dobře se prodávají v cizině .
OUT: prodávat dobře [lemma-"se",form-"se",agr-[case-aku]]
              [lemma-"cizina",form-"cizině",
               agr-[case-lok,gend-fem,num-sg],prep-"v"]
OUT: šplhat [lemma-"cena",form-"Ceny",
             agr-[case-nom,gend-fem,num-pl],prep-"BLANK"]
            [lemma-"stotisíc",form-"statisíců",
             agr-[case-gen,gend-inanim,num-pl],prep-"do"]
```

Figure 5: A sample output of AX. Given a sentence, the script first performs selection of simple sentences as described in section 3.2, then splits the sentence into individual clauses and moves the main verb to the beginning. (This type of output might be already used to automatically extract surface frames.) Sentences rejected during the analysis are not printed at all.

in treebanks, in order to gain sufficient coverage of Czech verbs. An approach of picking nice examples was described, including the filtering rules used for Czech and including the scripting language employed to express the rules.

Also, the two following steps of extracting verb frames were addressed. A simple linguistic filtration and hierarchical browsing of observed frames was suggested. The step from analytic (surface syntactic) description of verb frames to the full valency information is still more complicated. I was able to present not more than some warnings and known limitations. Both of these topics remain open for further research.

8 Acknowlegments

This article is based on my master thesis. I would like to thank to my supervisor, Vladislav Kuboň, PhD. This work was partially supported by the grant GAČR no. 201/02/1456 and by the grant GAUK no. 300/2002/A INF-MFF.

References

Böhmová, Alena, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2001. The Prague Dependency Treebank: Three-Level Annotation Scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers.

Bojar, Ondřej. 2002. Automatická extrakce lexikálně-syntaktických údajů z korpusu (Automatic extraction of lexico-syntactic information from corpora). Master's thesis, ÚFAL, MFF UK, Prague, Czech Republic. In Czech.

Brent, M. 1991. Automatic acquisition of subcategorization frames from untagged text. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 209–214, Berkeley, CA.

Briscoe, E. J. and J. Carroll. 1997. Automatic Extraction of Subcategorization from Corpora. In *Proceedings of the 5th ACL Conf. on Applied Nat. Lg. Proc.*, pages 356–363, Washington, DC.

Collins, Michael, Jan Hajič, Eric Brill, Lance Ramshaw, and Christoph Tillmann. 1999. A Statistical Parser of Czech. In *Proceedings of 37th ACL Conference*, pages 505–512, University of Maryland, College Park, USA.

Hajičová, Eva, Jarmila Panevová, and Petr Sgall. 1999. Manuál pro tektogramatické značkování. Technical Report TR-1999-07, ÚFAL MFF UK, Prague, Czech Republic.

Holan, T., V. Kuboň, K. Oliva, and M. Plátek. 1998. Two Useful Measures of Word Order Complexity. In A. Polguere and S. Kahane, editors, *Proceedings of the Coling '98 Workshop: Processing of Dependency-Based Grammars*, Montreal. University of Montreal.

Holan, T., V. Kuboň, K. Oliva, and M. Plátek. 2001. Complexity of Word Order. *Special Issue on Dependency Grammar of the journal TAL (Traitement Automatique des Langues)*, 41(1):273–300.

Korhonen, Anna, Genevieve Gorrell, and Diana McCarthy. 2000. Statistical Filtering and Subcategorization Frame Acquisition. In *Proc. of SIGDAT EMNLP and Very Large Corpora*, Hong Kong, China.

Kunze, J. 1972. *Die Auslassbarkeit von Satzteilen bei koordinativen Verbindungen im Deutschen*. Akademie-Verlag, Berlin.

Manning, Christopher D. 1993. Automatic acquisition of a large subcategorization dictionary from corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 235–242, Columbus, Ohio.

Panevová, Jarmila. 1980. *Formy a funkce ve stavbě české věty [Forms and functions in the structure of the Czech sentence]*. Academia, Prague, Czech Republic.

Panevová, Jarmila, Eva Hajičová, and Petr Sgall. 2001. Manuál pro tektogramatické značkování (III. verze, prosinec 2001). Technical Report TR-2001-12, ÚFAL MFF UK, Prague, Czech Republic. VS 960151, MSM113200006, NSF IIS-0121285.

Penn, Gerald. 2000. *The Algebraic Structure of Attributed Type Signatures*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University.

Sarkar, Anoop and Daniel Zeman. 2000. Automatic Extraction of Subcategorization Frames for Czech. In *Proceedings of the 18th International Conference on Computational Linguistics (Coling 2000)*, Saarbrücken, Germany. Universität des Saarlandes.

Sgall, Petr. 1967. *Generativní popis jazyka a česká deklinace*. Academia, Prague, Czech Republic.

Sgall, Petr, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Academia/Reidel Publishing Company, Prague, Czech Republic/Dordrecht, Netherlands.

Straňáková-Lopatková, Markéta. 2001. Homonymie předložkových skupin v češtině a možnost jejich automatického zpracování (Ambiguity of prepositional phrases in Czech and possibilities for automatic treatment). Technical Report TR-2001-11, ÚFAL/CKL, Prague, Czech Republic. In Czech.

Žabokrtský, Zdeněk, Václava Benešová, Markéta Lopatková, and Karolina Skwarská. 2002. Tektogramaticky anotovaný valenční slovník českých sloves. Technical Report TR-2002-15, ÚFAL/CKL, Prague, Czech Republic.

Žabokrtský, Zdeněk and Markéta Straňáková-Lopatková. 2002. Valency Dictionary of Czech Verbs: Complex Tectogrammatical Annotation. In M. González Rodríguez and C. Paz Suárez Araujo, editors, *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 949–956, Las Palmas de Gran Canaria, Spain. ELRA. CKL - LN00A063 (MŠMT).

Zeman, Daniel. 1997. A Statistical Parser of Czech. Master's thesis, ÚFAL, MFF UK, Prague, Czech Republic. In Czech.

Zeman, Daniel. 2002. Can Subcategorization Help a Statistical Parser? In *Proceedings of the 19th International Conference on Computational Linguistics (Coling 2002)*, Taibei, Tchaj-wan. Zhongyang Yanjiuyuan (Academia Sinica).