Annotation Lexicons: Using the Valency Lexicon for Tectogrammatical Annotation

Jan Hajič, hajic@ufal.mff.cuni.cz
Václav Honetschläger, honet@ufal.mff.cuni.cz

Institute of Formal and Applied Linguistics, Charles University, Prague
Center for Computational Linguistics, Charles University, Prague

Abstract

We present a formalization of the valency theory (Panevová, 1974) that fits the stratificational representation scheme used in the Prague Dependency Treebank. The notion of a lexicon as a repository of "static" (invariable, or context-independent) source of information is introduced; different type of lexicon is used at every layer of sentence representation, with a formal link to this representation (and thus, annotation).

In order to show how such a lexicon can be used in the annotation process itself, we describe also an automatic procedure using information from a valency lexicon for partial annotation of a corpus at the tectogrammatical layer. When adding nodes into tectogrammatical structure of sentences, we substantially increased recall at the cost of a small decrease of precision.

1 Motivation

The valency theory as a part of the theory of Functional Generative Description (Sgall, Hajičová, and Panevová, 1986) of language meaning has been around for some time (Panevová, 1974), yet it has never been properly formalized taking all complex interactions between morphology and syntax into account (a formalization of the FGD itself has been done by (Petkevič, 1995)).

The present article stems from the experience with annotation of the Prague Dependency Treebank, especially that of the highest (or deepest) tectogrammatical layer (for more details about the definition of the tectogrammatical layer itself, see (Hajičová, Panevová, and Sgall, 2000)). There is a valency lexicon that is being used during the annotation process, and there is a format definition which is currently being used (see also Sect. 4). However, the interaction of the valency lexicon (and for that matter, of all the other lexicons used in the annotation scheme, including the two lower layers) has not yet been fully formalized.

Moreover, as the annotation process continues, the partially created valency lexicon can be used to speed up the rest of the annotation process. We have developed an automatic procedure that uses information from the valency lexicon for preprocessing the data being annotated, saving some annotator's effort.

2 Introduction

2.1 The Prague Dependency Treebank

The Prague Dependency Treebank (PDT, `http://ufal.mff.cuni.cz/pdt`) is a long-term research project, whose main aim is a complex manual annotation of a (small) part of the Czech National Corpus.[1] It is being annotated on three layers.

On the lowest, *morphological layer* the lexical entry and values of morphological categories (person, number, tense, gender, voice, aspect, ...) are assigned to each word.

At the analytical and tectogrammatical layers, PDT is being annotated using a *dependency-based formalism.* At these layers a sentence is represented as a rooted tree. Edges represent relation of dependency (this relation is called "immediate subordination" in some other theories)—the edge is oriented such that it goes from the dependent node to its governor.[2]

At the *analytical layer*, a sentence is represented as a dependency tree described above. Nodes of the tree represent tokens (i. e. word forms and punctuation marks) as they are found in the original sentence. No node is added or deleted except for an extra root node, which is added. It has rather technical character and the predicate of a sentence depends on it. Edges usually (where it makes sense) represent relation of (reverse) syntactic control. In addition, several attributes are assigned to each node at this layer. The most important one is an *analytical function* capturing the type of dependency relation between the child and its parent, the other attributes being of rather technical nature.

The highest layer is the *tectogrammatical layer.* It captures the deep (underlying) structure of a sentence. Nodes represent only autosemantic words; synsemantic (auxiliary) words and punctuation marks are not represented by nodes, they may only affect values of attributes of the autosemantic words which they are attached to. Another principle is that if a unit is present in the meaning of a sentence but has not been expressed in it (on surface), a node representing this unit is added into the tectogrammatical representation of this sentence. Nodes are inserted mainly in the following two cases: when filling in an ellipsis with dependents actually present in the surface form of the sentence, and in the case when valency dictates so (see 5 and below). Edges represent relations between units of meaning. At this layer, several attributes are assigned to each node. One of the most important ones is the *(deep) functor* capturing the tectogrammatical function of a dependent relative to its governor.

At the analytical and tectogrammatical layer, coordinations are expressed in the following manner. Members of a coordination as well as modifications modifying all of them are children of the coordination node (which is usually a comma or a conjunction) and certain attributes of these nodes are used for distinguishing between these two types of children. Appositions are handled in a similar way.

For the purpose of this article, let us denote the three layers of representation as

- M for the lowest, morphological layer,
- A for analytical layer, and
- T for the tectogrammatical layer.

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

[2] Technically, we sometimes talk about a "parent – child" relation instead of a "dependent – governor" relation, since not all the edges represent linguistic relation of dependency as described above—some of them have rather technical character (e. g. edges from nodes representing punctuation marks).

There is also an *input* layer of tokenized text that will be denoted as "0".

Definition 1. A *layer* l is a member of a set of representation layers, L, where $L = \{0, M, A, T\}$.

The symbols 0, M, A, T will be used to index various other symbols to make sure the layers of representation are not confused. A lowercase l will be used as a variable when a certain definition or formula holds for more layers.

2.2 Formalization of Corpora and Lexicons

At every layer of representation we can identify units on which that representation is based. Informally, we can describe these units as being:

- a token (at the input and morphological layers), as determined by a tokenizer and the morphological processor, or
- a node of a dependency tree (at the analytical and tectogrammatical layers); the nodes at these two layers, however, differ substantially in their interpretation.

Definition 2. A *corpus* U_l (at a given representation layer $l \in L$) is a quadruple

$$U_l =< U, k, < V_j >_1^k, v > \tag{1}$$

where

- U is a set of abstract corpus *units* (sometimes called *positions*); there is always one distinguished unit, *nil* (which is typically used for "undefined" reference from the same or higher layer of representation).
- k is the number of *categories*[3] related to each unit $u \in U_l$, and thus the number of *labels* assigned to each unit in the corpus,
- $< V_j >_1^k$ is a set[4] of *lexicons* (V_j being the set of possible labels for j-th category), and
- v is a *labeling function*

$$v : U - \{nil\} \rightarrow \prod_{j=1}^{k} 2^{V_j} : v(u) =< v_1, v_2, v_3, ..., v_k >; \tag{2}$$

  j-th element of the complex label $v(u)$ will be denoted $v_j(u)$.

Furthermore, we require that the intersection of the sets U is pairwise empty across layers; for explicit reference to an item from the corpus quadruple, we introduce the following notation (naming convention):

$$UNITS(U_l) = U, \tag{3}$$

[3]Needless to say, the use of the term *category* here is not to be confused with Categorial Grammars.

[4]We will use the term *set* throughout this article for such n-tuples where in fact the order is not important, but for notational reasons it is easier to use the n-tuple notation anyway.

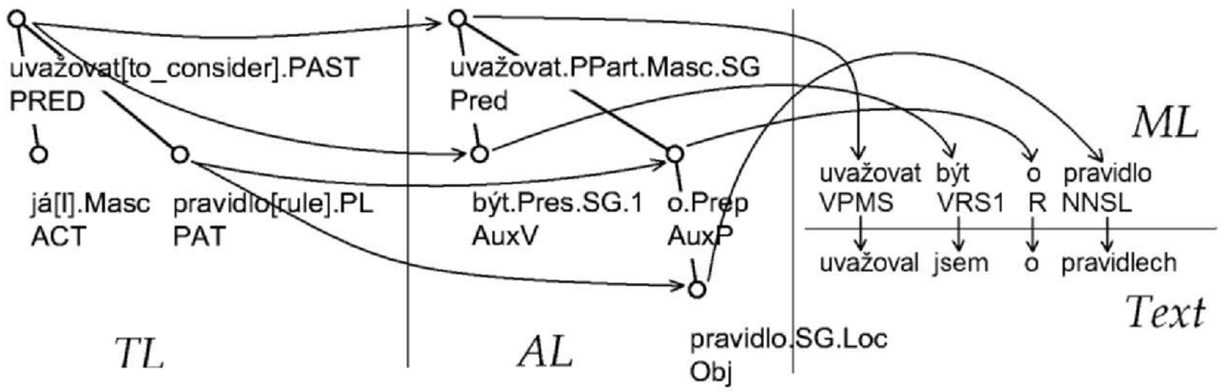

Figure 1: Links between units of annotation (*TL* – tectogrammatical layer, *AL* – analytical (surface syntax) layer, *ML* – morphological layer)

$$\#CATS(U_l) = k, \tag{4}$$

$$LEXICONS(U_l) = < V_j >_1^{\#CATS(U_l)}, \tag{5}$$

$$LABELS(U_l) = v. \tag{6}$$

It is natural to require that the lexicons V_j be finite, even though they might be of a vastly different nature. Linguistically interesting categories will typically be small sets, such as a tagset (for morphological features), or a set of analytical functions, or (large) sets of what is usually described as "lexical entries" in an NLP or other machine-readable dictionary. However, a labeling function v_j might also assign, for example, an integer denoting unit's corpus position(s)—even in such a case, the set V_j will be finite, namely the set of integers ranging from, say, 0 to $|U| - 1$.

In general, we can envisage categories with complex labels, such as n-tuples of strings or integers, or even (hierarchical) feature structures. For example, the categories representing NLP dictionaries will have such complex labels. Another kind of a category might be the set of all units at the same layer of representation, or even at a different layer[5].

Please note that the labeling function v—or more precisely, any of its components v_j—is defined as a mapping from the annotation unit set (U, Def. 2) to a product of a *subset* of V_j (Eq. 2). Such a generalization allows us to reuse this definition of a corpus for tiny corpora used "inside" lexicon entries (see below), and for ambiguous annotation.[6]

An example of the relation among layers of annotation is in Fig. 1. To simplify the picture, we have labeled the nodes by the values drawn from the appropriate lexicons, such as a tag lexicon, analytical function lexicon, lemma lexicon, etc.

[5]As opposed to the requirement that the corpus units be mutually different across layer, there is no such requirement for categories.

[6]There is no ambiguity in the manual annotation of the Prague Dependency Treebank, on any layer. However, we do acknowledge that sometimes it might be useful to allow for it, with proper semantics defined over the (ambiguity-allowing) annotation scheme. With the current definition of a corpus, we at least do not prevent this to happen in the future. Please note also that as the range of the labeling function v, we have used the product of subsets of the individual lexicons $(\prod_{j=1}^{k} 2^{V_j})$, not the subset of the product $(2^{\prod_{j=1}^{k} V_j})$—such a generalization would be *too* general and technically difficult to handle; so far, we have found we would need it only in very rare cases which we could circumvent with little pain, at least so far.

Formally, a lexicon can in general be defined as follows:

Definition 3. A lexicon V is a set of *entries* e having the form of an n-tuple (with n fixed for a given lexicon V) of *fields*:

$$V = \{e; e =< f_1, f_2, f_3, ..., f_n >\}, \tag{7}$$

where every field f_p ($p \in< 1..n >$) is either an *atomic value* (i.e., a value from some simple set S), or an n_p-tuple

$$f_p =< f_1, f_2, f_3, ..., f_{n_p} >, \tag{8}$$

where n_p ($= |f_p|$) is the size of the field tuple, and f_q ($q \in< 1..n_p >$) is a field.

In other words, every entry is a rooted tree with atomic values at its leaves[7]. Every entry has to have the same number of fields at the top level (i.e, the level just under the imagined root), but then the shape of the value tree may differ from entry to entry.[8]

In order to identify a particular value v in a lexicon entry (an atomic value as well as another non-atomic field), we will use the following notation:

$$v = f_{i_1.i_2.i_3...i_{max}}, \tag{9}$$

where i_d ($d \in< 1..max >$) is the position of the subtree to be traversed down at the current tree depth (i.e., tree depth d). The value of $f_{i_1.i_2.i_3...i_{max}}$ will be considered undefined in the obvious cases, i.e. when one of the indexes i_d falls out of the range $< 1..n_p >$ at a particular point of traversal.

For example, if for an entry e

$$e =< r, < x, y, z >, u >, \tag{10}$$

then[9] e.g. the following holds:

$$e_1 = r \tag{11}$$

$$e_{2.3} = z \tag{12}$$

and $e_{2.8}$ or $e_{4.1}$ is undefined.

The size of a particular tuple will be denoted in the usual way, e.g.

$$|e_2| = 3 \tag{13}$$

and we also define a $LEAF$ function

$$LEAF(e_i) = 1 \iff_{def} e_i \text{ is a leaf (and 0 otherwise).} \tag{14}$$

[7]This is a very simple yet very powerful data structure. It is also the only data structure we formally use here.

[8]In the following text, if we need to explicitly state that a variable v is of a type *field*, i.e., that it contains a tree-like structure of values, we say $Field(v)$.

[9]Remember the programming language LISP?

3 Lexicons in the Prague Dependency Treebank

Annotation is an ambiguous word: it could mean the *process of annotation* (whether manual or automatic) as well as the *result* of such a process (i.e. the annotated data). It is clear that there must be some knowledge encoded in something like a "lexicon" in order to automatically annotate a text.[10] For example, to do morphological analysis of Czech, one has to check whether in a given segmentation of a form to a root and an ending the two parts share a common inflectional class; classes are associated with roots and endings stored in lexicons.

It is less obvious what role can lexicons play in the annotated data: after all, annotation is meant to make implicit things explicit ("visible"), so why would we hide anything in lexicons? However, the reason to use lexicons[11] in annotated data is twofold: first, they make the annotated data reasonably small, since the information contained in them does not have to repeat at every occurrence of an annotation unit, and second, they make clear the difference between information that can vary position from position in the data and that we call *dynamic*, from the *static* (position-independent) information that is not influenced by the context of the particular unit and thus can be stored in a lexicon and only referred to from the annotated unit. If such a *reference* is a simple one that requires no processing other than retrieving the data (and it always should be so), one can even arguably consider the lexicon information explicitly present in the annotated data and their physical location becomes just a technical issue.

If we generalize little further, we can say that it is the references from annotation units to the lexicon entries—and only them—that constitute the annotation of the data. In the following sections, we define lexicons needed for the various annotation layers we use in the Prague Dependency Treebank. We have to admit, however, that in practice not all the actual annotation follows the formal definitions and constraints as described in the following sections; the treebank has been developed over a long period of time and only in the retrospective we could see more clearly the common ground on what the annotation is based (and make an attempt to generalize).

We should also stress that the following definitions are not directly related to the current annotation scheme itself (for its description, see (Hajič et al., 2001)), and probably will not be *directly* related to any future annotation scheme either[12]. However, it should serve as an underlying formal "theory" that can help when making the actual annotation scheme decisions.

3.1 Token Lexicons

Two lexicons are needed at the lowest layer of representation (layer 0): for tokens and positions[13]. The token lexicon is a simple list of strings consisting of the symbols of the alphabet of the particular language in question, and other symbols (possibly symbols from other languages, punctuation, special symbols etc.):

$$V_{token_0} = \{e; e =< s >; s \in A_0^*\}, \tag{15}$$

[10] Regardless what method is used, i.e. whether statistical or not; even statistical parameters must be eventually related to structures, words, phonemes, etc., so they can also be viewed as lexicons of sorts.

[11] The structure of these lexicons might be the same but also very different from the "lexicons" used for the annotation *processing*, depending again on the methods used.

[12] We are currently preparing a common, XML-based stand-off annotation scheme for the whole treebank, but that work is still very much in progress.

[13] The position lexicon is not really needed, since everything is in linear order and the index of the unit's position also defines its neighbors, but for uniformity with the higher layers, especially syntax, we introduce the ordering lexicon right here.

where A_0 is some alphabet[14].

The position lexicon is simply a set of all integers from 1 to the size of the corpus:

$$V_{position_0} = \{e; e =< i >; i \in< 1..|U| >\}, \tag{16}$$

where $|U|$ is a size of the corpus:

$$U_0 =< U, 2, < V_{token_0}, V_{position_0} >, v_0 > \tag{17}$$

where v_0 is a mapping defined for each unit $u \in U$ such that it assigns one position only to each u, it is unique[15] and dense[16] in the second dimension.

3.2 Morphological Layer Lexicons

The morphological layer contains no structure—it is still linear, with each token being at a particular position in the data. However, four formal lexicons are need for the full description of this layer, namely, to assign a *lemma* and a *tag* to every position of the input layer. Let the morphological-layer corpus be

$$U_M =< U, 4, < V_{lemma_M}, V_{position_M}, V_{tag_M}, V_{tokenref_M} >, v_M > . \tag{18}$$

Then the tag lexicon definition is similar to the token one:

$$V_{tag_M} = \{e; e =< s >; s \in T_M^+\}, \tag{19}$$

where T_M is some alphabet. However, the lemma lexicon is different: it contains all the morphological information necessary to get from a lemma to a form based on a tag:

$$V_{lemma_M} = \{e; e =<< t_1, f_1 >, ..., < t_g, f_g >>; t_i \in V_{tag_M}, f_i \in A_0^*\}, \tag{20}$$

For example, a lemma may contain the following pairs < VINF, *distribuovat* >, < VPP3S, *distribuuje* >, < VMMS, *distribuoval* > (for the infinitive form, present 3rd pers. sg. form, and past participle form in masc. anim.) and many more (for those not familiar with Czech, these are some of the surface forms of the verb *distribuovat*, lit. *to distribute*).

Each entry in the lemma lexicon thus encodes a mapping function from tags to forms[17].

The position lexicon is defined similarly to the token one:

$$V_{position_M} = \{e; e =< i >; i \in< 1..|UNITS(U_M)| >\}, \tag{21}$$

[14] In practice, we do not limit the alphabets A_l beyond natural constraints required by the technical markup used.

[15] We do not want two units to share a position.

[16] Although not critical, we also do not want "holes" in the position numbering to make our lives easier when defining neighbors.

[17] In an implementation, the lemma is often represented as a string, corresponding to a word base form, and the lexicon is indexed by this string for better human readability; in the above example, the string *distribuovat* would probably be chosen as such a descriptive label. Also, the morphological information is often (and the PDT is no exception) encoded in some more compact way, but at the level of abstraction used in this article it is sufficient to use the above defined simple mapping function.

and the token reference lexicon links the lemma unit to the input token(s):

$$V_{tokenref_M} = \{e; e =< u_1, ..., u_r >; u_i \in UNITS(U_0)\}, \tag{22}$$

The number of links in a single entry is always one in the Prague Dependency Treebank ($r = 1$ for every entry, since tokenization already introduces sentence boundaries and thus there is an extra token for every sentence break at the input token layer). In general, we cannot exclude the possibility that a lemma comes from two or more forms (such as fixed collocations, or, should we deal with e.g. with several kanji characters considered tokens at the input layer, etc.). In such cases, $r > 1$. We also do not exclude the case that $r = 0$ (an empty tuple in the lexicon entry)—it can be used when the morphological unit comes from "nowhere"[18]. On the other hand, it is quite imaginable that there is a single input token referenced from several lemma units; again, this does not happen in the Prague Dependency Treebank[19] but even Czech displays phenomena that could reasonably be represented in this way: e.g., the contracted forms *nač, oč, naň* consisting of a preposition and a pronoun, the clitic *-s* representing the auxiliary verb *jsi* in 2[nd] person sg. past tense, etc.

In the manually annotated portion of the Prague Dependency Treebank, the size of every of the four components of v_M is one, i.e. the is no ambiguity in annotation.

3.3 Analytical Layer Lexicons

In principle, only two things are being annotated at the analytical layer: a *dependency* and an *analytical function* (dependency relation label). In addition, the original word order[20] has to be preserved (or at least accessible) in order not only to "draw" the trees correctly, but to be able to restore the original token order of the morphological (and consequently, the token) layer.

Let the analytical-layer corpus be

$$U_A =< U, 3, < V_{wordref_A}, V_{govref_A}, V_{afun_A} >, v_A > . \tag{23}$$

Let further U_0 and U_M be the corresponding token-layer and morphological-layer corpora, respectively.

The *dependency* is annotated by simply referring to another node (within the same sentence). In the realm of lexicons as defined in 2.2, we need a lexicon of references to nodes at the same (i.e., analytical) layer:

$$V_{govref_A} = \{e; e =< u >; u \in UNITS(U_A)\}, \tag{24}$$

The analytical function is a simple label, and the lexicon is formally not much different from a tag lexicon V_{tag_M} (except for an alphabet, in general):

$$V_{afun_A} = \{e; e =< s >; s \in A_A^+\}, \tag{25}$$

[18] This can be used for units denoting sentence boundaries, ellipsis resolution when it is solved at the morphological layer, etc. Again, none of this happens in the Czech-language Prague Dependency Treebank.

[19] With the exception of the words *aby, kdyby*, nevertheless the technical solution adopted there does not really use such links.

[20] As annotated at the morphological layer, that is. It might not be identical to the token-layer order. In the PDT, it is, with the unimportant exception of *aby, kdyby* as discussed in the previous section.

Finally, we need the backward links to the morphological-layer annotation. In our definition of the analytical layer, there can only be one link per analytical unit; in other words, it embodies the principle that "every token gets a node" (Hajič et al., 1997):

$$V_{wordref_A} = \{e; e =< u >; u \in UNITS(U_M)\}, \tag{26}$$

As can be seen from the above definitions, the analytical layer is in fact simpler than the morphological layer. This is in accordance with the need to define nothing more than dependency relations on top of the (morphologically analyzed) words. However, it does not at present contain any cross-layer lexicon relating the analytical-layer units with the morphological ones.[21]

Despite the fact that none of the analytical-layer lexicons needs the notion of a dependency tree, it is natural to introduce a definition of a dependency tree here, since we will deal with it throughout the rest of the paper.

Definition 4. Let $U'_D =< U', 1, < V'_{govref_D} >, v' >$ be a corpus, $V'_{govref_D} \subset U'$. A unit $d \in U'$ is a *descendant* of a unit $u \in U'$ iff there exists a $(g+1)$-tuple $< u_0, ..., u_g >$ such that $u_0 = d$, $u_g = u$, and for all $0 < i \leq g$, $u_i \in v'_1(u_{i-1}), |v'_1(u_{i-1})| = 1$. U'_D is called an *dependency tree* iff there exists exactly one unit $r \in U'_D$ for which $v'_1(r) = \emptyset$ and for all $u \in U'_D, u \neq r$, u is a descendant of r and for all $u, d \in U'_D$ holds that if d is a descendant of u, u is not a descendant of d.[22]

The above definition simply states that a dependency tree is a tree in the sense of graph theory (see e.g. (Aho, Hopcroft, and Ulman, 1974)), using the notion of a corpus as defined earlier. Typically, U'_D is a very small "corpus", corresponding to one original sentence. Let us now relate the above general dependency tree definition to a (real) corpus:

Definition 5. Let $U'_D =< U', 1, < V'_{govref_D} >, v' >$ be a dependency tree. U'_D is an *out-of-corpus tree* (relative to a corpus $U_l =< U, k, < V_j >_1^k, v >$) iff $U \cap U' = \emptyset$. Analogically, U'_D is a *corpus tree* (belonging to a corpus $U_l =< U, k, < V_j >_1^k, v >$) iff $U' \subset U$ and there exists j $(1 \leq j \leq k)$ such that $V_j \subset U$ and for every $u \in U' - \{nil\}, v'_1(u) = v_j(u)$.

We need this definition to distinguish between trees that are part of a corpus (such as the analytical-layer corpus) and trees that are not part of a corpus (such as trees stored in lexicons, which we will need later).

Since we typically have some sort of linear ordering defined on the nodes of a dependency tree (at least for corpus trees), we can formally define the notion of projectivity:

Definition 6. Let $U'_D =< U', 1, < V'_{govref_D} >, v' >$ be a dependency tree, and o an ordering function $U' \to \mathcal{N}$.[23] We say that U'_D is *projective* (relative to o) iff for all $u, d \in U'_D$ holds that if $u \in v'_1(d)$ then for all $w \in U'_D$ such that $min\{o(u), o(d)\} < o(w) < max\{o(u), o(d)\}$, w is a descendant of either u or d.

Let us stop here with the addition of analytical-layer labeling to the general definition of a dependency tree:

[21] Such as the lexicon V_{lemma_M} does at the morphological layer, where it relates the lemmas and tags with token-layer forms.

[22] Ignoring the requirement that $|v'_1(u_{i-1})|$ be 1, we would get a (limited) *dependency forest*. For the purpose of this article, we do not want to complicate things too much and thus we will always work with single dependency *trees*.

[23] $\mathcal{N}$ being a set of all integers, naturally.

Definition 7. Let $U'_A =< U', 3, < V'_{wordref_A}, V'_{govref_A}, V'_{afun_A} >, v'_A >$ be an analytical-layer corpus. We say that U'_A is an *analytical-layer dependency tree* iff $< U', 1, < V'_{govref_A} >, v'_{A_2} >$ is a dependency tree.[24]

It is desirable that a ("real") analytical-layer corpus is a collection of dependency trees, and a modification of the original definition of an analytical-layer corpus to fulfill this requirement is only a technical matter.[25] Analytical-layer corpus dependency trees are not in general projective, since we typically use the morphological-layer token order[26], as defined by the composition of the analytical-to-morphological back-reference function v_{A_1} and the position-defining v_{M_2} labeling function of the morphological-layer corpus.[27] Out-of-corpus trees do not necessarily have an associated ordering function defined, and so it might not be possible to define projectivity on them.

In the manually annotated portion of the Prague Dependency Treebank, the size of every of the three components of v_A is one, i.e. there is no ambiguity in annotation.[28]

3.4 Tectogrammatical Layer Lexicons

The tectogrammatical layer is the most complex layer of annotation reached so far, and thus the structure of its lexicons is also quite complex.

The tectogrammatical layer annotation can be subdivided into four areas:[29]

1. Tectogrammatical dependency structure covers the (labeled) *dependency* relations at the tectogrammatical layer (dependency labels are caller *functors* to distinguish them from analytical *functions*). The units are autosemantic words and some necessary technical nodes (for handling coordination, for example). Therefore, the number of units of the tectogrammatical dependency tree in general differs from the number of units of the same sentence at the analytical layer.

2. Topic/focus annotation includes labeling of every node of the tectogrammatical dependency tree by topic/focus/contrastive topic, as well as reordering of the nodes using so-called *deep word order*.

3. Co-reference consists of linking certain nodes of the tectogrammatical tree to another nodes, denoting co-reference relations (both grammatical and textual, to a limited extent). There might be several types of co-reference relations, represented by different types for these links.

4. Grammatemes are relatively simple node labels for information not directly derivable from elsewhere (*unit grammatemes*), or secondary dependency labels (*syntactic grammatemes*) for more detailed distinctions of functors.

[24] The notions of a *descendant*, *out-of-corpus tree* and a *corpus tree* in an analytical-layer dependency tree are defined analogically using the same projection to a general dependency tree.

[25] However, to avoid too much technicalities here, we leave such a modification to the reader.

[26] In Czech, this is in fact the original word order.

[27] We also have to properly technically define the composition, since the values of v_{A_1} are sets.

[28] For those familiar with the PDT, we would like to stress that the use of the "double" analytical functions `AtrAdv`, `AtrObj`, etc., does not in fact constitute an ambiguity (not even technically) – without going into much detail, it is a representation of a systematic "double-dependency".

[29] In practice, the manual annotation was also organized along lines corresponding to these four areas.

In the present article, we will only deal with a simplified structure, ignoring for the moment the details of coreference, and topic/focus annotation, as well as part of so-called grammateme annotation.

In any case, we want again to have the tectogrammatical-layer units linked back to the analytical layer, and there is a *valency lexicon* that contains complex structure of such (possible) links, in a manner similar to the way inflections are stored in the V_{lemma_M} lexicon. However, since there is no corresponding lexicon at the analytical layer, we will have to use pointers to information located at the morphological layer, too.

Let the tectogrammatical-layer corpus be

$$U_T =< U, 7, < V_{aref_T}, V_{govref_T}, V_{func_T}, V_{tfa_T}, V_{coref_T}, V_{gram_T}, V_{val_T}, >, v_T > . \tag{27}$$

Let further U_0, U_M and U_A be the corresponding token-layer, morphological-layer and analytical-layer corpora, respectively.

The entries of the back-reference lexicon are (not surprisingly) defined as sets,[30] since the unit correspondence does not have to be 1:1:

$$V_{aidref_T} = \{e; e =<< u_1, ..., u_r >>; u_i \in UNITS(U_A)\}, \tag{28}$$

We allow $r = 0$ for newly created units (such as for restored ellipsis). In practice it often happens at the tectogrammatical layer of the Prague Dependency Treebank that $r \neq 1$.

The formal definition of the lexicon for the "dependent $\rightarrow$ governor" links corresponds to the definition of the dependency lexicon at the analytical layer; in other words, this lexicon simply contains a set of all governing nodes at the tectogrammatical layer across the whole corpus:[31]

$$V_{govref_T} = \{e; e =< u >; u \in UNITS(U_T)\} \tag{29}$$

The lexicon for functors contains an extra field for functor's syntactic grammatemes:

$$V_{func_T} = \{e; e =< s_f, s_g >; s_f \in A^+_{s_f}, s_g \in A^*_{s_g}\} \tag{30}$$

An empty string s_g denotes an undefined syntactic grammateme. It is allowed for functors that do not require further distinctions[32].

The topic/focus annotation contains an assignment of the unit to topic, focus, or contrastive topic relative to its parent, and also the deep word order position reference:

$$V_{tfa_T} = \{e; e =< b, dord >; b \in \{t, f, c\}, dord \in < 1..|U| >\} \tag{31}$$

Now let us define a tectogrammatical-layer dependency tree (cf. the Def. 7 for counterpart definition of an analytical-layer dependency tree):

[30] We are using an n-tuple in the definition, since we have limited ourselves to n-tuples in the only allowed data type of a field (cf. Eqs. 7 and 8). We simply ignore the order of the references in this case. See also footnote 4.

[31] In practice, we require that no node has its governor outside a current sentence. We are leaving the issue of sentence boundaries outside the scope of this article, but it is not difficult to formally define them and subsequently, we could easily define constraints based on them.

[32] There is a mapping that assigns each value of a functor a set of possible syntactic grammatemes, including none; see (Hajičová, Panevová, and Sgall, 2000) for a complete list.

Definition 8. Let $U'_T =$

$$< U', 7, < V'_{aref_T}, V'_{govref_T}, V'_{func_T}, V'_{tfa_T}, V'_{coref_T}, V'_{gram_T}, V'_{val_T}, >, v'_T >$$

be a tectogrammatical-layer corpus. We say that U'_T is a *tectogrammatical-layer dependency tree* iff $< U', 1, < V'_{govref_T} >, v'_{T_2} >$ is a dependency tree.[33]

Contrary to the analytical-layer corpus, we require that the tectogrammatical-layer dependency trees be projective relative to the *dord* ordering (cf. the v_{4_2} function, Eqs. 27, 31 and 2).

There are two types of coreference[34]: a regular coreference and a complement-type coreference[35]:

$$V_{coref_T} = \{e; e =< u_r, u_c >; u_r \in UNITS(U_T), u_c \in UNITS(U_T)\} \tag{32}$$

Grammatemes are values that accompany the structure, the topic/focus annotation and the coreference annotation in order to keep the meaning of the original sentence preserved (such as number, degree of comparison etc.). We will not go in much detail here, though, simply denoting the number of unit grammatemes as G:

$$V_{gram_T} = \{e; e =< g_1, ..., g_G >; g_i \in A^*_{g_i}\} \tag{33}$$

Before we can define the overall structure of the valency lexicon, we have to define several substructures. Every time we mention dependency trees that are part of a lexicon, we (naturally) mean out-of-corpus trees (Def. 5).

Informally, a valency *frame* is a single correspondence between a one-level deep, tectogrammatical-layer dependency tree (the "function", cf. (Sgall, Hajičová, and Panevová, 1986)) and a corresponding analytical-layer dependency tree (the "form")[36]. It is considered rooted in the "current" unit (which is important when matching it with a subtree in the annotated corpus). In general, we require that every unit at the tectogrammatical layer contains such a frame (even though in many cases this correspondence will only be trivial).

Therefore, with a fully annotated tectogrammatical dependency tree and the correspondences from the valency lexicon at hand, we can (in an ideal case, with no ambiguity present in the function-form relation) to construct the structure of the appropriate analytical tree. However, we cannot reconstruct the full annotation; most notably, the analytical (and thus, the token) order cannot be determined directly. Similarly, certain morphological tags (or rather their "parts") cannot be determined solely on the basis of the tectogrammatical structure and the valency lexicon, the other attributes have to come to play as well (especially the grammatemes from V_{gram_T}).

In order to combine information from the various sources available in the tectogrammatically annotated tree when relating it to the analytical one (for its reconstruction in generation, for example), we must allow certain information in the tectogrammatical $\leftrightarrow$ analytical correspondence (concerning mostly the labeling of its analytical-layer dependency trees) within the

[33] The notions of a *descendant*, *out-of-corpus tree* and a *corpus tree* in an tectogrammatical-layer dependency tree are defined analogically using the same projection to a general dependency tree.

[34] An annotation unit can have both. There is also another typology, namely, the (exclusive) distinction between grammatical and textual co-reference, but that one is irrelevant for the discussion here.

[35] Let us remind you here that such a reference might point to "nothing" by setting the pointer to the *nil* unit of $UNITS(U_T)$; not to be confused with the "undefined" value equal to an empty set, which is not used in the manual annotation). As expected, for most tectogrammatical units in the Prague Dependency Treebank, both of the references are *nil*.

[36] Of a possibly different depth.

valency frames be underspecified. Moreover, we cannot[37] work with the analytical layer only; the repertoire of its functions is not rich enough to encode everything for the corresponding analytical $\leftrightarrow$ morphological relation. Thus we in fact relate the tectogrammatical layer to *both* the analytical *and* morphological layer at once.

Definition 9. Let U_T be a tectogrammatically annotated corpus (with corresponding U_A, U_M and U_0 annotations; see Eqs. 17, 18, 23, 27 for their contents and notation). A *valency frame* f is a sextuple

$$f =< S, L, O, M, F, C >, \tag{34}$$

where

S is a set of *slots*

$$S = \{s\}, \tag{35}$$

L is a *slot-labeling function*

$$L : S \rightarrow V_{func_T} \text{ (cf. Eq. 27)}, \tag{36}$$

O is an *optionality flag*

$$O : S \rightarrow < 0..p > \tag{37}$$

(a slot s being *optional* in a *degree* $O(s)$ iff $O(s) > 0$, otherwise it is *obligatory*[38]),

M is a set[39] of size $k \geq 0$ of morphological-layer corpora

$$M =< U'_{M_i} >_{i \in <1..k>}, \tag{38}$$

where

$$U'_{M_i} =< U''_i, 4, < V_{lemma_M}, V'_{position_M}, V_{tag_M}, \{0\} >, v'_{M_i} >, \tag{39}$$

F is a set[40] of analytical-layer dependency trees

$$F =< U'_{A_i} >_{i \in <1..k>}, \tag{40}$$

where[41]

$$U'_{A_i} =< U'_i, 3, < \{u; u \in U''_i\}, V'_{govref_A}, V_{afun_A} >, v'_{A_i} >, \text{ (cf. Def. 7)}, \tag{41}$$

[37] At the moment, although we would like to arrive at a “pure” two-layer-only relations in the future.

[38] Let us leave the semantics of optionality aside for the moment. Look at it just as a function mapping the slots to integers, as the definition does. Of course, the optionality influences the logic of the match between a lexicon and an annotated corpus.

[39] See footnote 4.

[40] See footnote 4.

[41] For every i, U'_{M_i} is the morphological-layer corpus corresponding to the tree U'_{A_i} in the sense mentioned right after Eq. 23.

and

C is a k-tuple of *slot-mapping functions*

$$C = < C_1, C_2, ..., C_k > \tag{42}$$

such that for every i, $1 \leq i \leq k$,

$$C_i : U'_i \to S, \tag{43}$$

where for every $s \in S$ there is a $u \in U'_i$ such that $C_i(u) = s$.[42]

Slots are considered rooted in a single node (i.e., the corresponding annotation units depend on a common unit which is a root of such a (sub)tree).

The analytical-layer trees U'_{A_i} ($\in F$) and especially their corresponding morphological-layer "corpora"[43] will have many values of their labeling functions $v'_{A_{i_3}}$ and (all four components of) v'_{M_i} undefined (i.e., their value will be the whole lexicon) or underspecified (their value will be a proper subset of the lexicon of size greater than 1); we will call such underspecified analytical trees "constraints" because they effectively describe a (possibly large) set of (fully specified) analytical trees. Let us stress here again that these trees are not part of the annotated corpus (they are out-of-corpus trees, cf. Def. 5; we have only mentioned the corpora U_T, U_A, U_M and U_0 in the above definition because we needed to use the appropriate lexicons, but the annotation itself is "physically" separate.[44]

As can be seen from the whole definition of the valency frame, there is no "word" or other human-readable and intuitively understandable identification being associated with it. The reason is that analogically to the morphological-layer corpus, where a "lemma" is represented rather as a mapping between tags and surface letter (or, phoneme) sequences, tectogrammatical slots are gradually mapped also all the way to the sequences of phonemes (through the analytical-layer and morphological-layer constraints), and that is all what is needed. Again, analogically to the morphological-layer, in the actual implementation and annotation we do identify the valency frames by various identification strings, both technical and human-readable, such as *to believe* for the valency frames that eventually appear as forms of the verb *believe*, etc.

A *valency lexicon* can then be defined as

$$V_{val_T} = \{e; e = << f_1, ..., f_r >>\}, \tag{44}$$

where f_i is a valency frame. We will postpone the lexical-semantic interpretation of a valency frame to some future work; there is a working hypothesis that essentially there is at least one lexical meaning per valency frame, and that we need two or more different valency frames per a single lexical meaning only very rarely and in well defined circumstances (Sgall, 2003). In fact, we could easily adapt the above definition in a way of further generalization that would fully accommodate the "one meaning $\to$ one frame" hypothesis, but it is not quite clear what advantage it would have (and it would make the definition even clumsier).

[42] The mapping goes, as the definition says, from the analytical-layer units (nodes) to the slots, solely for technical reasons (we can avoid another subset-based function range in the definition). Traditionally, however, this relation is viewed in the "generation" direction as slots (functions) being mapped to surface forms, thus the name, a *slot-mapping* function.

[43] These "corpora" are of course just short sequences of lemmas and tags, corresponding to the analytical-layer tree.

[44] That's why we have so many "primed" variables in the above Def. 9.

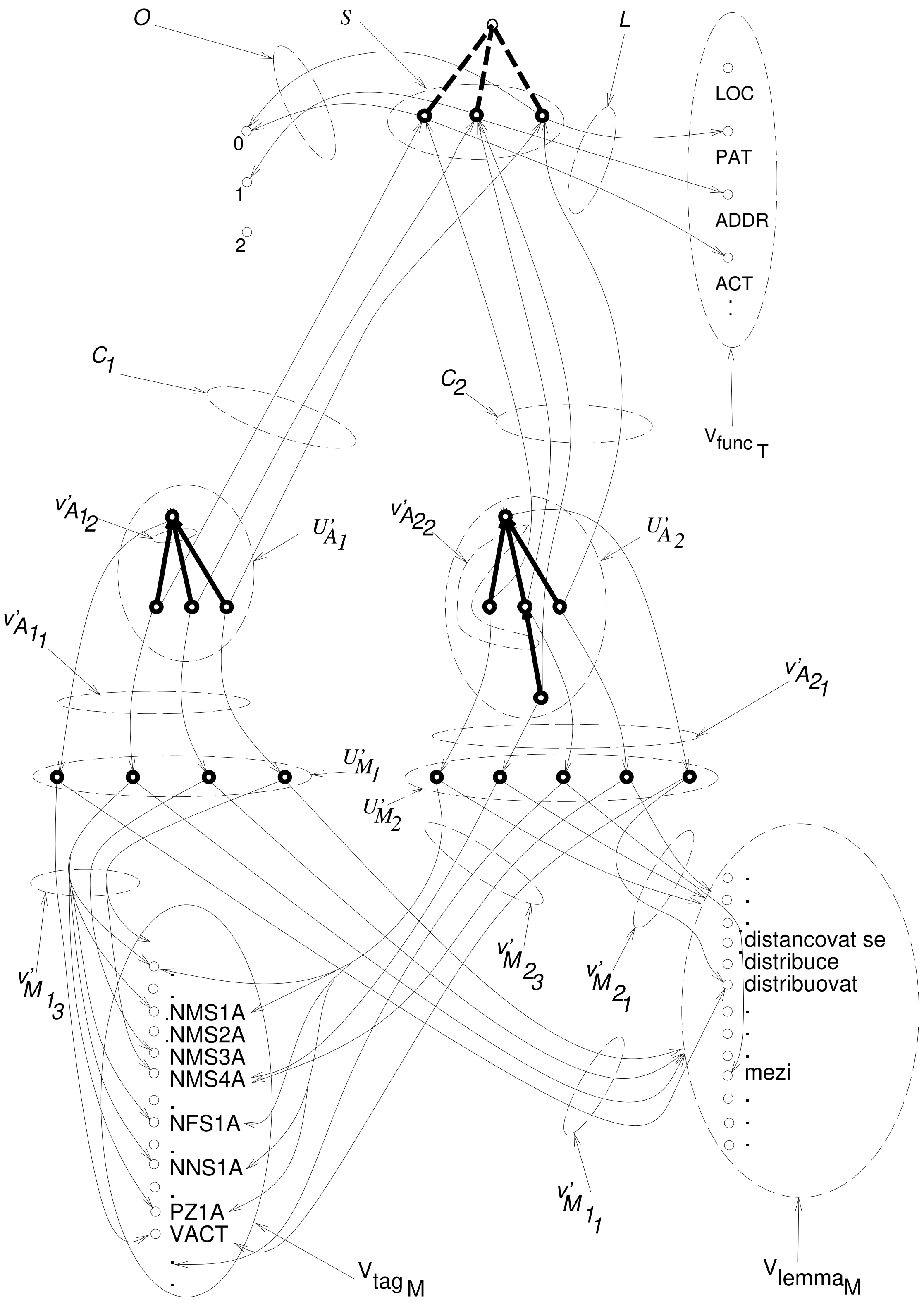

Figure 2: Structure of the valency frame for *distribuovat* (lit. *to distribute*) (cf. Def. 9).

Graphically, a valency lexicon can be visualized as in Fig. 2: it contains an example of (a part of) a valency frame for the verb *distribuovat* (lit. *to distribute*). The markup in the figure corresponds to Def. 9;[45] for example, L is the labeling function which in this case assigns the functors ACT, PAT and ADDR to the three valency slots (top part of the figure), two of them are obligatory (ACT and PAT) and one optional (ADDR, cf. the mapping O, also in the top part of the picture). Only two forms (analytical-layer dependency trees) are used (cf. the middle part of the picture): one (U'_{A_1}) merely copies the structure and sets the following constraints on the morphological surface form:

- the lemma of the analytical-layer dependency tree root must be *distribuovat*,[46] and it must be in active mood,
- the case of the node corresponding to the ACT-labeled slot must be in nominative (marked as "1" in the abbreviated tags; simplifying the situation a bit, we can say that we do not care about the values of other morphological features, such as gender, part of speech, etc.),
- the case of the node corresponding to the PAT-labeled slot must be in accusative ("4"; again, regardless of other morphological features,
- the case of the node corresponding to the ADDR-labeled slot must be in dative ("3", and again, regardless of other morphological features.

The other one (U'_{A_2}) changes the constraints for the surface analytical-layer and morphological-layer form. First of all, the analytical-layer tree is two-levels deep, and there are two nodes linked to the slot labeled ADDR: the "middle one" and its dependent. Following the morphological-layer links, the corresponding constraints can be tracked down for this two nodes:

- the node immediately dependent on the node with the morphological lemma *distribuovat*[47] must be a preposition *mezi*[48] (lit. *in between*), and
- the lowest node (its dependent) must be in the accusative case ("4"; again, regardless of other morphological category values).

Obviously, it would be very difficult to create and edit the valency frames in the form depicted in Fig. 2; please recall that we are presenting the underlying formalization here, not the actual markup as used in the valency lexicon(s) we use. Based on the experience with possible surface forms of most valency frames, we could identify several patterns of the correspondence between slots and their surface forms, which allows us to dramatically simplify the presentation of the valency frames.

First of all, we can consider the surface forms of valency slots within a single valency frame in most cases as mutually independent. Combined with the observation that they almost always depend on the root of the corresponding analytical-layer dependency tree, and that the analytical-layer units correspond to the morphological-layer ones always[49] 1:1, we can in fact describe the surface form of the slots using a simple correspondence

[45] Leaving out some less important sets and mappings, such as the set V_{afun_A} and it's corresponding mappings, and more importantly, abbreviating the lemma lexicon (Eq. 20) to its entry labels in the sense of footnote 17, for space-saving reasons.

[46] See footnote 17.

[47] See footnote 17.

[48] See footnote 17.

[49] No wonder: it has been a design principle to do it so (Hajič et al., 1997).

$$\text{slot name} \qquad (\text{alt.form}_1,\text{alt.form}_2,\ldots,\text{alt.form}_n)$$

Obviously, we can use slot names (as assigned by the slot-labeling function L) directly in the presentation, and we have also developed various shortcuts for the constraints on morphological form: for example, to represent a single analytical-layer node with a corresponding set of morphological tags (a subset of V_{tags_M}) that should contain the accusative case value (while everything else being essentially irrelevant) we simply write "4" as the alt.form$_i$.[50] The type of bracketing that surround the alternative forms determines optionality: parentheses denote an obligatory slot, square brackets an optional one. Analytical-layer dependency tree root lemmas are typically the same in all forms, thus it is necessary to include them only once. Therefore, the above valency frame can be presented as

distribuovat[51] ACT(1) ADDR[3,*mezi*+4] PAT(4)

Facing such a simple presentation, a natural question arises why we have ever bothered with a relatively complicated formalization of the notion of a valency frames (and lexicons in general). The answer is simple: the above form of presentation, albeit it works for most cases, does not work in general. However, since this form has traditionally been used for a long time, and due to its brevity, we will use it in the rest of this paper when describing our tool(s) for annotation support regarding valency frames, abandoning the description of handling the more difficult valency frames for a moment.

An informal account on the relation of cross-layer links (such as the functions v_{M_4} (Eq. 18) and v_{T_2} (Eq. 27)) and lexicons (V_{lemma_M} and V_{val_T}, respectively) can be found in (Hajič and Urešová, 2003), with more concrete examples (mostly from the valency lexicon).

We stop here in the attempt to formalize the structure of lexicons in the PDT. More constraints can be put to those lexicons, especially the valency lexicon. For the rest of the article, these constraints will be described informally, since full formalization of these constraints has not been developed yet.

4 Valency in the Prague Dependency Treebank

In linguistics the term "valency" indicates the capability of lexical units to bind other terms onto itself; their number and character is determined. Valency properties of words differ to such an extent that when we want to have information about valency of words, we have to describe them one by one, i. e. to create a *valency lexicon.*

The information contained in a valency lexicon is thought believed to be able to help when solving various tasks from the area of natural language processing (adopted from (Lopatková et al., 2002)). These are lemmatization, morphological tagging, syntactical analysis, and word sense disambiguation. All of the aforementioned tasks has to be solved by a computer when it performs e. g. a machine translation. Another task which information from a valency lexicon can help with is building other linguistic resources. In this article we deal with this task and suggest how an automatic procedure for partial annotating a corpus based on a valency lexicon can operate.

[50] Other obvious shortcuts have also been used, such as "p" for possessive adjectives, "preposition+case-number" for a double-node analytical-layer tree representing a preposition and its dependent in particular morphological case, "vv" for a general relative clause, etc.; we also leave the lemma constraint completely out if it is in fact the whole V_{lemma_M}.

[51] See footnote 17.

In the framework of FGD, valency theory has at first been developed for verbs (see (Panevová, 1974)); then also for some other parts of speech (nouns and adjectives). We deal with valency of verbs only here.

Expressions which can modify a verb are called *modifications* (in a dependency tree, the verb is a governor for its modifications and they are its dependent nodes). Modifications are divided into *inner participants* and *free modifications*.

Inner participants are such modifications for which the following criteria hold: they can occur at most once (without being in a coordination or apposition) as a modification of the verb in question; and they cannot occur as a modification of every verb. FGD distinguishes (and in the PDT and the lexicons are used) five inner participants: actor (ACT), patient (PAT), addressee (ADDR), origin (ORIG), and effect (EFF).

Free modifications are such modifications for which the following criteria hold: they can occur multiple times with a certain verb; and they can modify every verb (possible constraints have rather content character, not a grammatical one). List of all the free modifications used for annotating PDT (and thus of those used in the lexicons) is in (Hajičová, Panevová, and Sgall, 2000).

Both inner participants and free modifications can be (in particular verb and its meaning) either *obligatory* (for the given verb and its meaning its obligatory modification has to be present in tectogrammatical structure of every sentence where this verb occurs) or *optional* (it does not need to be there). However, even an obligatory modification does not need to be expressed in a surface form of a sentence, e. g. in Czech we can reply to a question just with a bare verb. The criterion deciding whether a modification is obligatory or optional is the so called *dialogue test* ((Panevová, 1974)).

A *valency frame*[52] contains inner participants and the obligatory free modifications of a verb in its particular meaning. Every verb (in the lexical sense) has at least one valency frame and each frame usually corresponds to one meaning of a verb.

5 The Valency Lexicons and Annotation

In an ideal world, we could simply extract all the valency frames for all (meanings of) verbs from the manually annotated corpus at the tectogrammatical layer. The annotators would not deal with any lexicon, they would simply annotate occurrences of verbs and their modifications filling in missing nodes and labeling them accordingly, based on the definition of valency. However, this is not feasible for several reasons: the definition above (as well as the intuition behind it) is rather vague; and not all occurrences of verbs (or their meanings) are encountered in the relatively small manually annotated corpus. From the first reason it follows that the consistency of the annotation will be inferior, while the (small) size of the corpus does not allow us to rely on the statistics to filter out the “noise”. For this reason annotators need a valency lexicon since we believe that it will substantially improve the consistency of annotation.

At the beginning of tectogrammatical annotation (i. e. the highest layer annotation, in which the analytical layer structures are “converted” and enriched to become the tectogrammatical structures) no suitable valency lexicon existed. As stated above there are two valency lexicons being developed. Development of both of them started with the same intention—to help annotators—however they differ in the approach to their creation and extensions.

[52] More precisely, the set S of a valency frame, cf. Def. 9.

The lexicon being created by annotators during annotation (called PDT-VALLEX, see (Hajič et al., 2003)) captures only those meanings (and thus those frames) of verbs which occur in the annotated material. It also contains examples of the usage of frames. It currently contains 4457 verbs (as well as 1425 nouns and 21 adjectives).

VALLEX, the other lexicon described in detail in (Lopatková et al., 2002), captures all the meanings of the verbs contained in it. Besides valency frames it also captures syntactically relevant features of verbs, e. g. reflexivity, reciprocity, and control. It also contains examples of usage of frames, synonyms of verbs and their classes etc. It currently contains 1102 verbs with 3333 frames.

The initial version of PDT-VALLEX was generated from the VALLEX data. However, the content of the two lexicons has not been since synchronized for their annotation schemes differ in several aspects.

Valency frames captured in VALLEX contain extra types of modifications—so called *quasi-valency modifications* (described as modifications with semantics of free modifications and some features of inner participants) and *typical optional free modifications*; for a detailed discussion see (Lopatková et al., 2002). Annotators also include extra modifications into PDT-VALLEX, e. g. typical free modifications.

5.1 Morphosyntactic Forms

If we constrain the correspondence function C in the definition of a valency frame (cf. Def. 9) in such a way that the analytical-layer dependency trees F can be segmented just under its root into continuous dependency trees, each corresponding to a single slot from S, the (surface) morphosyntactic forms can attached to the slots directly. There are three degrees of optionality (optional/quasi/typical) in VALLEX, and one (optional) in PDT-VALLEX.

Morphosyntactic form of a modification can generally be described by the following:

- the case expressed as the number,
- preposition, character "+", and the case expressed as the number,
- symbol "inf" for the infinitive form of a verb,
- subordinating conjunction (for subordinated clause with a conjunction),
- symbol "vv" (for subordinated clause without a conjunction),
- part of a phraseme.

Since the set of morphosyntactic forms is not frozen we can marginally see their other shapes in both lexicons.

Here is an example of valency frames of verb *jednat* {*be in treaty/proceed/treat*} (adapted from VALLEX), it is enriched with synonyms and examples of usage.[53] English translations are in braces.

jednat

ACT(1) ADDR(*s*{*with*}+7) PAT(*o*{*about*}+6)

[53]The notation is adapted from PDT-VALLEX. A slot is noted down as its functor plus list of its morphosyntactic forms in parentheses (when the modifications is obligatory) or in brackets (otherwise).

– synonym: *vyjednávat* {*transact*}
– example: *jedná s nimi o investicích* {*he is in treaty with them about investment*}

ACT(1) MANN[]

– synonym: *konat* {*proceed*}
– example: *začal jednat* {*he started to proceed*}

ACT(1) PAT(*s*{*with*}+7) MANN( )

– synonym: *zacházet* {*treat*}
– example: *jedná s ní špatně* {*he treats her badly*}

5.2 The Current Status of the Tectogrammatical Annotation

Annotation of PDT is performed at three layers. First, the text is annotated on the morphological layer, then on the analytical layer, and finally on the tectogrammatical layer.

Tectogrammatical annotation is performed semi-automatically. Basic rules for it are described in (Hajičová, Panevová, and Sgall, 2000), however, refinement of the annotation instructions as well as a development of tools for partial automatization of the annotation process are still in progress. Tectogrammatical annotation consists of three parts—the first one is automatic, the second one manual, the third one automatic again.

The first phase is an automatic procedure whose input are manually annotated structures. These structures are taken as the basis of the tectogrammatical structures being created. It performs the following tasks:

- determines the value of the functor in clear-cut cases,
- determines the values of several attributes (e.g. modalities and aspect for verbs and the sentence modality),
- hides most nodes of synsemantic words and fills the corresponding attributes of the nodes which they depend on accordingly (e.g. subordinating conjunctions, prepositions, and punctuation marks);
- reattaches nodes in certain cases (and thus adjusts the tree structure).

These tasks are performed by a tool called AR2TR ((Böhmová, 2001)), which is run in batch mode (offline) before the annotators get the data in their hands.

The second phase is manual. The human annotators

- deal with the remaining synsemantic words,
- adjust the resulting tectogrammatical structure,
- add nodes for words not expressed on the surface,
- enter and/or check the values of attributes, including the most important one—the functor.

There also exists a statistical, decision-tree-based tool for assigning functors at the tectogrammatical layer, see (Žabokrtský, 2001). In this paper we call it AFA. This tool is supposed to be used once the tectogrammatical structure is fully and correctly determined; however, annotators report that even if used at the beginning of their work (i. e. right after AR2TR) it makes their work more productive. In either case, they have to correct values assigned by it.

The third phase is automatic and it primarily determines some additional values of attributes of nodes added in the previous phase.

5.3 The Goal of Development of Our Tool

A tool using information from a valency lexicon can help with the tectogrammatical annotation in two ways. First, it can determine functors of modifications of a verb as required by its valency frame(s) based on morphosyntactic forms of these modifications. Second, when it determines that an obligatory modification of the verb is missing it can add it to the tectogrammatical structure—this is the second way.

With respect to the annotation procedure described above it seems natural to apply this tool at the very beginning of the second phase.

6 The Algorithm

In this part we describe the algorithm that performs the task described above.

6.1 The Core of the Algorithm

The algorithm uses the initial tectogrammatical structure as created by AR2TR and the valency lexicon. Obligatory modifications are the most "interesting" from our goal point of view: for a given verb (more precisely, its meaning) an obligatory modification *has to be present* in the *tectogrammatical representation* of every sentence where this verb occurs. An optional modification need not be present in the tectogrammatical representation, but it does contain valuable information about the surface morphosyntactic form of the modification which in turn can help us to determine the correct functor for such a modification. However, even an obligatory modification does not need to be expressed in the *surface form* of a sentence (in an extreme case one can, e.g., reply to a question just with a bare verb with no modifications). This fact, i.e. the possibility of not seeing some of the modifications expressed in the original sentence, is what makes the task non-trivial.

Moreover, the valency frames corresponding to the individual meanings of the verb usually overlap; however, it is impossible to choose the correct frame first[54] and then simply deal only with the slot-to-modification alignment. Instead, we align, match and score all possible frames and try to put together pieces of information from those ones with the maximal score. The match score is based on the alignment of the (form of) possible modifications as found in the text with the morphosyntactic form(s) associated with slots in a valency frame from the lexicon. This measure is designed in such a way that it has two desirable properties: (1) when no modification is expressed in the text, the scores of all the frames are equal; (2) when there is only one frame with all modifications present, such a frame has the highest score.

The algorithm works as follows.

[54] Not having (yet) a good Czech WSD, that is.

1. Get the morphosyntactic forms of modifications (as they appear in the data).

2. For every lexicon frame of the verb compute the alignment (and from it the match score) between slots of this frame and the modifications present in the sentence (using the surface morphosyntactic forms). Retain only the frame(s) with the maximum score.

   e. g. frame: ACT(1) PAT(4) ADDR[3] MEANS[7]
   expressed modifications: 1 (nominative), 3 (dative), 4 (accusative), v+6 (preposition *v* {*in*} with locative)
   alignment: ACT, ADDR, PAT, none
   the score (total number of matches) is 3

3. Assign functors to the modifications according to the computed alignment. If more than one frame is retained, assign functors according to all such frames. Assign no functor to a modification if there are conflicting functors (but treat all these conflicting functors as if they were assigned).

   e. g. verb *připravit* {*prepare/steal*} has two frames
   ACT(1) PAT(4) for *prepare*
   ACT(1) ADDR(4) PAT(*o*+4) for *steal*
   expressed modifications: 1 (nominative), 4 (accusative)
   matches: ACT, none (PAT/ADDR conflict)

4. Add such nodes (with appropriate functors) to the tree that are not present in the tree but they are matched as obligatory in (all of the) frame(s).

5. Assign the rest of the functors determined by the alignment.

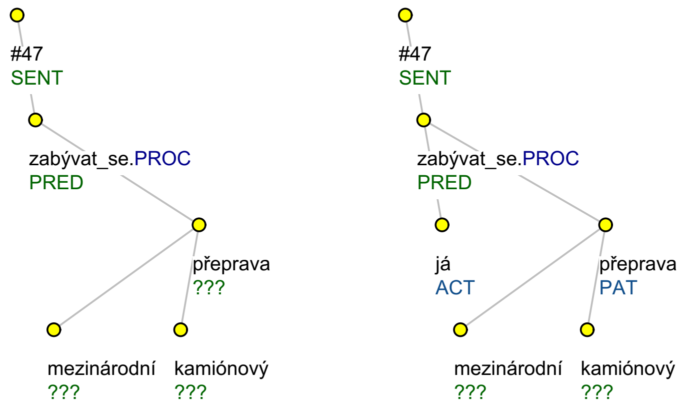

Figure 3: An example of the input (on the left) and the output (on the right) of our tool applied to the sentence *Zabývám se mezinárodní kamiónovou přepravou.* {*I am engaged in international truck transportation.*}

6.2 Additional Details of the Algorithm

When several modifications of a verb are coordinated (and they may be coordinated recursively—being in a coordination of coordinations etc.) they have to be considered as one modification for the purposes of alignment and match with possible valency frames. Our tool solves it in the way that the common morphosyntactic form of such a coordination is the unification of forms of the coordinated modifications. We cannot claim the morphosyntactic forms to be identical because not all the pieces of information are the same in all the coordinated modifications. For example in *pro Petra a tebe* {*for Peter and you*} the form of the first member is *pro*{*for*}+4 and that of the second one is 4. However, even unification does not solve all problems, for example in *Naskládal nákup do kufru a na sedačku.* {*He stacked up the shopping into the boot and onto the seat.*} forms to be unified are *do*{*into*}+2 and *na*{*onto*}+4 and their unification is impossible. Naturally when the functor is determined it has to be assigned to all members of coordination.

Another case is when the verb itself is a member of coordination (again, it may be in a coordination of coordinations etc.)—then all modifications common for members of this coordination have to be (and are) considered as its modifications. Appositions are handled the same way as coordinations.

The problem occurs when a modification of a verb in coordination should be added into the tectogrammatical structure. This modification may be common for all coordinated verbs (in case that all the verbs can have this type of modification) and then it should be attached to the coordination node instead of to the verb. We have to decide which alternative is true. We consider this problem hardly resolvable and demonstrate it with the following example:

> *Q1: Zavolal Pavel Janě, nebo ji navštívil?* {*Did Paul call Jane or did he visit her?*}
>
> *Q2: Zavolal Pavel Janě, nebo ji Petr navštívil?* {*Did Paul call Jane or did Peter visit her?*}
>
> *A: Nezavolal a nenavštívil.* {*He-did-not-call and he-did-not-visit.*}

In Czech we had to add node(s) representing the actor(s) *on* {*he*} into the tectogrammatical tree representing the answer. However, if the first question has preceded this answer, both clauses in the answer have this actor common and it should be attached to the conjunction *a* {*and*}. If the second question has preceded it, there should be two nodes for two (different) actors depending on particular verbs. Whenever possible the modifications are considered to be common.

Our tool can handle verbs in the passive voice. In the Czech language, an actor, generally in nominative, takes on an instrumental case when passivized; a patient, generally in accusative, takes on an nominative case. The tricky part is determination whether a (tectogrammatical-layer) verb was expressed in active or passive voice—this information is not contained directly in analytical nor in tectogrammatical structures. (Since verbs in passive voice are compound, the analytical layer is too low for explicit capturing this information; and from the point of view of the tectogrammatical layer the voice which a sentence has been expressed in is uninteresting because its meaning is the same in either case.) So our tool has to analyze compound verb forms found at the analytical layer to determine the voice of a verb.

There has been a need to handle expressions with counted objects correctly. In Czech there are prepositional constructions which affect case of both numerals and counted nouns (e. g. *o dvaceti lidech* {*about twenty people*}); and other which affect only case of numerals; counted nouns are in genitive (e. g. *pro dvacet lidí* {*for twenty people*}), this is true also for the case when the whole phrase is in nominative. At the analytical layer these cases are represented by the following structures. In the first case to the preposition there is attached the counted object and the numeral is attached to it. In the second case to the preposition there is attached the numeral and the counted object is attached to it. At transition to the tectogrammatical layer the preposition is reattached to its child and is hidden and its child is reattached to its former grandparent. So the proper handling of expressions with counted objects is such that when we encounter a noun without a preposition and with a numeral as its child, we should determine the morphosyntactic form of modification corresponding to it from this numeral, not from the noun itself.

We have to assign several attributes of nodes added by our tool to the tectogrammatical layer, mainly their generated lemma, gender, and number have to be filled. The rules describing their values are stated in (Hajičová, Panevová, and Sgall, 2000) and our tool implements them.

There can occur the situation when our tool is about to assign one functor representing an inner participant several times (through a valency frame found in the lexicon). This is inadmissible, hence our tool assigns this functor to neither modification, since we do not know which one to assign it to.

7 Experiments and Results

We have made series of experiments concerning modifications of our algorithm, usage of different valency lexicons, utilize of already available information from the tectogrammatical layer, and cooperation with some other programs used during tectogrammatical annotation.

7.1 Test Data and Evaluation

Test data consists of 1641 both analytically and tectogrammatically manually annotated sentences. On the tectogrammatical layer, these data contain 18620 nodes coming from the analytical trees (7040 of them are modifications of verbs) and 2229 nodes newly added into the tectogrammatical trees (1211 of them are modifications of verbs). Since valency frames in PDT-VALLEX are updated during tectogrammatical annotation, only data more recent than the used version of PDT-VALLEX have been used to ensure fair evaluation.

We report results divided into two separate groups according to operations which our tool performs: it adds nodes into the tectogrammatical structure and assigns functors to nodes coming from the analytical trees. Precision (P; in per cents), recall (R; in per cents), and F-measure (F) of those groups of operation are reported. A node is considered to be added correctly iff it is attached to the correct node and its functor is determined correctly. We compute recall as the ratio of the correctly added nodes (or the correctly assigned functors) to all the added nodes in the data (or to all the nodes coming from analytical trees and retained in tectogrammatical trees). We compute precision as the ratio of the correctly added nodes (or the correctly assigned functors) to all the nodes added (or the functors assigned) by our tool.

Since our tool is and always will be applied after the AR2TR tool, we always report cumulative results obtained by serially applying both tools. Of course, we also report results of adding nodes by the AR2TR tool alone.[55]

Unless stated else we use PDT-VALLEX for our experiments.

[55]We report them only for operations common for AR2TR and our tool; as stated before, it also can determine values of some other attributes, hide nodes, and reattach them.

7.2 The Basic Experiments (Table 1)

In the first row there are results of application of AR2TR alone and in the second one those of subsequent application of our tool based on the described algorithm. In the following rows there are results of application of our tool enhanced by the following features.

(a) The match between modifications appointed by certain frame and the expressed modifications is computed using obligatory modifications only.

(b) Conflicts of functors corresponding to a modification are solved by random selection of one of the conflicting functors. (We recall that *none* functor was being assigned initially.)

(c) In the valency lexicon, there are sometimes no constraints on the morphosyntactic form of a frame slot; therefore our tool could not assign the appropriate functor. In this case the forms extracted from (Hajičová, Panevová, and Sgall, 2000), where lists of possible functors for several morphosyntactic forms are defined, are used.[56]

(d) In PDT-VALLEX there are valency frames of not only verbs, but also of nouns and adjectives. We tried to use them as well.

(e) When a verb was not found in the lexicon, the default frame containing the only modification—obligatory actor expressed by a nominative case—is assigned to it.

Table 1: Results of the basic experiments

	Functors			**Nodes**		
Experiment	P	R	F	P	R	F
AR2TR alone	93.4	18.8	31.3	86.7	17.3	28.8
basic implementation	93.0	28.0	43.0	67.5	48.3	56.3
match according to obligatory (a)	93.0	28.5	43.6	67.4	48.2	56.2
random functor when conflict (b)	92.6	29.5	44.7	69.5	48.5	57.1
extracted morphosyntactic forms (c)	92.3	28.7	43.8	69.0	48.4	56.9
valency frames of all POSs (d)	90.4	33.8	49.2	63.7	48.8	55.3
default frameset (e)	93.9	28.3	43.5	68.2	49.4	57.3
the “final” method (a)+(b)+(c)+(e)	92.8	30.9	46.4	72.5	49.5	58.8

Based on the results of these experiments, we have incorporated features (a), (b), (c), and (e) into our tool and we report further results using them.

7.3 Experiments with Various Valency Lexicons (Table 2)

Two lexicons were created on basis of VALLEX and PDT-VALLEX: PDT-VALLEX/VALLEX is a copy of PDT-VALLEX plus those verbs with their frames of VALLEX not present in PDT-VALLEX; analogically with VALLEX/PDT-VALLEX. All four lexicons have been evaluated with our tool while keeping everything else intact.

According to our expectations when using VALLEX instead of PDT-VALLEX precision increased and recall decreased (VALLEX is hand-checked, contains more meanings of individual verbs, but contains less entries). When we use PDT-VALLEX/VALLEX instead of PDT-VALLEX we obtain the same results since almost no verbs from VALLEX that are not already

[56] From those, only those forms corresponding to prepositional phrases have been used.

Table 2: Results of experiments with various valency lexicons

Experiment	Functors			Nodes		
	P	R	F	P	R	F
PDT-VALLEX	92.8	30.9	46.4	72.5	49.5	58.8
VALLEX	92.9	20.5	33.6	78.6	44.0	56.4
PDT-VALLEX/VALLEX	92.8	30.9	46.4	72.5	49.5	58.8
VALLEX/PDT-VALLEX	92.1	30.6	45.9	70.1	49.3	57.9

in PDT-VALLEX appear in the test data. We conclude that using PDT-VALLEX is the best possibility.

7.4 Experiments with Usage of Functors Already Assigned (Table 3)

We made several experiments concerning the usage of functors already assigned by AR2TR.

We tried to solve conflicts of functors corresponding to a modification in the way that when there is some functor already assigned by AR2TR, we do not overwrite it (thus the random selection is done only if there is no functor there).

When the alignment between modifications appointed by certain frame and the expressed modifications is computed and this alignment would not assign the same functor as this already assigned by AR2TR to a modification, this frame is effectively discarded (its match is set to 0).

When aligning lexicon frames with morphosyntactic forms the functors already assigned by AR2TR are regarded as correctly assigned and are not subject of alignment. The alignment is computed only on the other functors and forms.

Table 3: Results of experiments with usage of functors already assigned

Experiment	Functors			Nodes		
	P	R	F	P	R	F
current best result	92.8	30.9	46.4	72.5	49.5	58.8
when conflict, do not overwrite	92.8	30.9	46.4	72.5	49.5	58.8
discard non-alignable frames	92.0	30.3	45.6	65.4	49.6	56.4
regard assigned functors as correct	92.9	30.7	46.1	72.8	49.3	58.8

None of these features has been involved into our algorithm.

7.5 Experiments on Cooperation with AFA (Table 4)

As stated before, annotators apply AFA (a tool assigning functors) during the second phase of tectogrammatical annotation. The goal of the experiments described in this section has been to determine the optimal mutual order of application of AFA and of our tool as well as to decide whether these tools should or should not overwrite functors assigned by their predecessors. The results obtained by application of AFA alone are also presented. Only results of assignment of functors are tabbed since AFA does not change the given structure.

Table 4: Results of experiments on cooperation with AFA

Experiment	**Functors**		
	P	R	F
our tool non-overwriting alone	91.9	30.6	45.9
our tool overwriting alone	92.8	30.9	46.4
AFA non-overwriting alone	88.4	84.5	86.4
AFA overwriting alone	88.3	84.9	86.6
our tool non-overwriting + AFA non-overwriting	88.5	85.3	86.9
our tool non-overwriting + AFA overwriting	88.4	84.5	86.4
our tool overwriting + AFA non-overwriting	88.8	85.6	87.2
our tool overwriting + AFA overwriting	88.4	84.5	86.4
AFA non-overwriting + our tool non-overwriting	88.3	85.1	86.7
AFA non-overwriting + our tool overwriting	88.8	85.6	87.2
AFA overwriting + our tool non-overwriting	88.4	84.9	86.6
AFA overwriting + our tool overwriting	88.8	85.2	87.0

We can draw conclusions that the variant slightly superior to the other is to use AFA in non-overwriting mode and our tool in overwriting mode regardless their mutual order.

8 Closing Remarks

We have presented an attempt to formalize the annotation scheme and the lexicons used in the Prague Dependency Treebank project. It might seem to some as an intellectual exercise only; after all, we do have a formal markup scheme (SGML/XML) for both the annotation and the lexicons. However, we have encountered various problems in interpretation of the markup, and thus we feel that a full formalization might help.

We have also tried to ease the manual annotation process of PDT using information from valency lexicons, with positive results (F-measure gain of 30.0 over a baseline for node insertion, and minimal gain for functor assignment), and also subjectively positive human feedback. Former version of the tool has been used in the project of machine translation (see (Hajič et al., 2003)).

There are several ways of improving the quality of our tool. One of them is a complete understanding of the information contained in the valency lexicons—our tool cannot handle compound prepositions and phrases as well as some special cases of the morphosyntactic forms (e. g. the constraint that a modification have to be an adjective).

9 Acknowledgments

This research was supported by a grant of the Grant Agency of the Czech Republic No. 405/03/0913 and a project of the MŠMT ČR No. LN00A063.

References

Aho, A., J. Hopcroft, and J. Ulman. 1974. The Design and Analysis of Computer Algorithms.

Böhmová, Alena. 2001. Automatic Procedures in Tectogrammatical Tagging. *The Prague Bulletin of Mathematical Linguistics*, 76.

Hajič, Jan, Yuan Ding, Jason Eisner, Martin Čmejrek, Terry Koo, Kristen Parton, Gerald Penn, Drago Radev, and Owen Rambow. 2003. Natural Language Generation in the Context of Machine Translation, Workshop '02 Final Report. Technical report, CLSP Technical Reports, Baltimore, MD, U.S.A.

Hajič, Jan, Eva Hajičová, Petr Pajas, Jarmila Panevová, Petr Sgall, and Barbora Vidová Hladká. 2001. The Prague Dependency Treebank on CDROM. Linguistic Data Consortium, Univ. of Pennsylvannia, Philadelphia, PA. ISBN 1-58563-212-0.

Hajič, Jan, Jarmila Panevová, Eva Buráňová, Zdeňka Urešová, and Alla Bémová. 1997. A Manual for Analytic Layer Tagging of the Prague Dependency Treebank. Technical Report TR-1997-03, ÚFAL MFF UK, Prague, Czech Republic. in Czech.

Hajič, Jan, Jarmila Panevová, Zdeňka Urešová, Alevtina Bémová, Veronika Kolářová, and Petr Pajas. 2003. PDT-VALLEX: Creating a Large-coverage Valency Lexicon for Treebank Annotation. In *Proceedings of TLT 2003*, Växjö, Sweden.

Hajič, Jan and Zdeňka Urešová. 2003. Linguistic Annotation: from Links to Cross-Layer Lexicons. In *Proceedings of TLT 2003*, Växjö, Sweden.

Hajičová, Eva, Jarmila Panevová, and Petr Sgall. 2000. A manual for tectogrammatic tagging of the prague dependency treebank. Technical Report TR-2000-09, ÚFAL MFF UK, Charles University, Prague.

Lopatková, Markéta, Zdeněk Žabokrtský, Karolína Skwarska, and Václava Benešová. 2002. Tektogramaticky anotovaný valenční slovník českých sloves. Technical Report TR-2002-15, ÚFAL/CKL MFF UK, Charles University, Prague.

Panevová, Jarmila. 1974. On verbal Frames in Functional Generative Description. *Prague Bulletin of Mathematical Linguistics*, 22:3–40.

Petkevič, Vladimír. 1995. A New Formal Specification of Underlying Representations. *Theoretical Linguistics*, 21:7–61.

Sgall, Petr. 2003. Personal communication.

Sgall, Petr, Eva Hajičová, and Jarmila Panevová. 1986. The Meaning of a Sentence in Its Semantic and Pragmatic Aspects.

Žabokrtský, Zdeněk. 2001. Automatic Functor Assignment in the Prague Dependency Treebank. Technical Report TR-2001-10, ÚFAL MFF UK, Charles University, Prague.