Building a Linguistically Interpreted Corpus of Bulgarian: the BulTreeBank

**Kiril Simov, Petya Osenova, Milena Slavcheva,
Sia Kolkovska, Elisaveta Balabanova, Dimitar Doikoff,
Krassimira Ivanova, Alexander Simov, Milen Kouylekov**

BulTreeBank Project
http://www.BulTreeBank.org
Linguistic Modelling Laboratory, Bulgarian Academy of Sciences
Acad. G. Bonchev St. 25A, 1113 Sofia, Bulgaria
kivs@bgcict.acad.bg, petyaosenova@hotmail.com, milena@lml.bas.bg,
sia@ibl.bas.bg, elisavetabal@yahoo.com, dim_doikoff@abv.bg,
krassy_v@abv.bg, adis_78@dir.bg, mkouylekov@dir.bg

Abstract

In the field of Human Language Technology (HLT), the existence of linguistically interpreted real-world texts provides the license necessary for a given language to enter the area of high-tech applications. The significance of BulTreeBank is the granting of an HLT license to a "less processed" language like Bulgarian which, until recently, has been formally modelled and processed mainly on the morphology level. The BulTreeBank project aims at the creation of syntactically annotated data for Bulgarian and the tools for their production, management and automatic processing. It provides not only language resources, but develops an infrastructure of research solutions, production scenarios and services.

1. Introduction

In recent years the existence of syntactically interpreted corpora has become a crucial prerequisite for the development of the linguistic theories and the variety of NLP application tasks. Our treebank project (Simov, Popova and Osenova, 2001) relies on the following methodological assumptions:

- **Fine-grained description.** The linguistic descriptions in the treebank are aimed to be rather detailed in order to demonstrate the multilayered information flow in the syntactic structure of the sentences. Thus not only constituent labels, but dependency relations and grammatical functions have been added.
- **Theory dependency.** On more detailed levels of linguistic granularity it becomes necessary to exploit some theory-dependent components, which assure the consistency of the representation.
- **Formal representation.** Robustness on the level of linguistic descriptions is achieved by the formal representation of the annotation scheme elements. In our opinion, the notion of formal representation refers to the fact that the annotation scheme has a logical interpretation, which allows for inference and check for inconsistency of the annotated data. Our formal mechanism is a special representation of HPSG grammars and sentence analyses as feature graphs (see (King and Simov, 1998) and (Simov, 2001)) based on a logical formalism for HPSG (see (King, 1989)).
- **Partial analyses and predictions.** The detailness of the descriptions, the size of the treebank and the desired high level of quality requires semi-automatic facilities for the annotators during the annotation process. Such a minimization of human labour is achieved maily by exploiting all the possibilities for providing automatic partial analyses of the input string before the stage of the 'attachment-resolution' annotation.

In order to implement this methodology, we rely on an interface between the constraint-based linguistic theory and the principles of shallow parsing.

The theory. We have chosen Head-driven Phrase Structure Grammar (HPSG) (see (Pollard and Sag, 1987) and (Pollard and Sag, 1994)) as an underlying theory of our treebank due to the following reasons:

- it allows for a simultaneous representation of the constituent and dependency relations, combining the tree structure with the feature graph encodings.
- it also allows for a consistent description of linguistic facts on every linguistic level. Thus, it ensures an easy incorporation of the linguistic information, which does not belong to the level of syntax.
- The formal basis of HPSG enables non-problematic translation to other formalisms.
- The universal principles of HPSG can be used to support annotators' work during the development of the treebank.

The shallow parsing output. The results of the partial processing stage have to be compatible with the HPSG setup of the annotation in the treebank. This means that the output structures together with the assigned grammatical information correspond to the HPSG sort hierarchy and yield HPSG principles. This integration is ensured by a mapping, which transforms the morpho-syntactic positional tagset into an HPSG-compatible one. It means that more fine-grained infor-

mation is added concerning the head-nonhead dependency relations and the grammatical functions. Then these specifications are further incorporated into the chunk level phrases.

Parsing with a grammar. Usually the whole process of annotation is considered a continuous subsequence of steps from the tokenization stage to the full syntactic analysis. However, taking into account the fact, that we use a symbolic grammar as a parsing scheme, we first concentrated on the named entity recognition task, aiming at initial linguistic interpretation of the text elements. By this approach the coverage and robustness were improved. Similar approach was explored for LFG-based annotation scheme (Dipper, 2000).

The structure of the paper is as follows: Section 2 outlines the structure of the BulTreebank corpus and describes the specific language data areas in it. In Section 3 the architecture of the annotation process is presented. Section 4 describes the XML supporting software tool - the CLaRK system.

2. Linguistically Interpreted Language Data

The BulTreeBank corpus consists of the following subsets of linguistically interpreted language data: 1) a core set of sentences; 2) a treebank; 3) a text corpus.

- **Core set of sentences.** Supplemented with detailed syntactic descriptions, the core set of sentences is intended to serve as a test-suite for software applications processing Bulgarian texts on the syntactic level. The core set consists of 2500 sentences extracted mainly from Bulgarian grammar books: thus they illustrate main classes of language phenomena. Since the core set consists of manually analysed sentences, it is used as a "golden standard" within the BulTreeBank project. From one side it is the reference manual for the human annotators at different steps of their intervention in the production chain, and on the other side it is a test bed for the grammar rules applied in the different automated processes.

  The basic linguistic phenomena covered in the core set of sentences belong to the broadly defined cross-linguistic classes of phenomena like: sentence and clause types, predication, verb complementation, diathesis, agreement, modification, modality, tense and aspect, word order, coordination, negation.

  The descriptions attached to the sentences contain the specific manifestations of the phenomena in Bulgarian. For instance, there are very rich paradigms of synthetic and analytic verb forms which interact specifically with pronominals and particles and produce discontinuous segments according to the language specific linearization principles. Bulgarian is a pro-drop languange in subject position, it allows object doubling and multi-functional verbal clitics. A somewhat exotic feature is the inflection-like definite article.

- **Treebank.** The treebank is a set of syntactically annotated sentences excerpted from naturally occurring Bulgarian texts. The foreseen size of the treebank is one million words. The sentences are selected regarding two factors. Firstly, they are supposed to fit into the classification of syntactic structures in Bulgarian as defined in the core set of sentences. In case phenomena are discovered which are new to the classification in the golden standard, they may be added to it. Secondly, the sentences in the treebank indicate the statistical distribution of the types of syntactic structures defined in the core set of descriptions. Having an evaluation of the frequency of occurrence of types of syntactic phenomena, the treebank sentences can be grouped in frequency classes. The exact methodology of constructing the treebank will be worked out after a series of experiments.

  In the course of building the treebank, special attention will be paid to resolving ambiguity problems in the context of use of the sentences. This can be done by including in the treebank not only isolated sentences, but also whole paragraphs or even a number of consequent paragraphs. When sentences are annotated in context, a description of the disambiguating context will be added to the syntactic description. At present we envisage an informal description of the context to be inserted in the syntactic annotation of sentences. An attempt will also be made to formalize the context description, in case we have the opportunity for that.

- **Text corpus.** The text corpus is a collection of different types of texts having wide domain coverage. Its size is 100 million words, which is, generally speaking, the recommended size of a national corpus. The text files are converted into XML format and contain multi-layered linguistic annotation added incrementally to the text data.

  The first layer is paragraph level TEI conformant encoding of the logical structure of the texts. The second layer is that of morphosyntactically analyzed word tokens. The subsequent layers of encoding of linguistic information is achieved by the development and ordered application of partial grammars. They are formulated as regular expressions and fulfil the following tasks: sentence boundary delimitation; recognition of numerical expressions and special symbols; recognition of proper names and foreign words; recognition of abbreviations and attachment of their full names. The tricky moment in the partial analyses is in the ordering of applications of the partial grammars, since the language phenomena that are processed are interrelated and the decisions taken for a given application order of the partial grammars define their interoperability scenarios. Tests are carried out for measuring the efficiency of different grammar ordering. At all stages of processing appropriate TEI conformant markup is added.

3. The Architecture of the Annotation Process

The syntactic annotation process within BulTreeBank relies on the development of two formal grammars: 1) a partial grammar for shallow parsing; 2) an HPSG-based general grammar.

1. **Partial grammar for shallow parsing.** We compiled several reliable regular-expression chunk grammars for automatic identification of the non-recursive phrases. They are applied in a cascade order. For example, AP chunks and PP chunks are identified after the NP chunk recognition. Thus AP chunks contain only the elements, which are not a part of an NP chunk and PP chunks are formed by combining a preposition with a following NP. In accordance with Abney's ideas (see (Abney, 1991), (Abney, 1996)) about 'easy-first parsing' and 'islands of certainty', we relied on the presence of clear indicators for specifying the chunk boundaries. For example, the adverb is not a clear indicator for the beginning of an NP, unless it is trapped between a preposition and a noun.

2. **HPSG grammar.** This grammar incorporates three elements: universal principles, language specific principles for Bulgarian and a lexicon. The HPSG grammar component is used in the project as a general constraint over the possible output syntactic structures of one and the same input sentence. The constraint is activated during the editing stage of the annotation process, checked by human experts. We also envisage to use information from it as a top-down filtering over the partial grammars. It is encoded into an HPSG grammar development tool, ensuring the appropriateness conditions within the generated analyses.

The linguistic knowledge, encoded in the HPSG sort hierarchy, forms the backbone of the syntactic descriptions of the Bulgarian data. The sort hierarchy defines all possible linguistic structures that are further constrained by the grammar and/or the information, entered by the annotators. The annotation schemata, employed during the project, allow for composite tag definitions. Thus each tag in the syntactic structure, being decomposable, ensures the distribution of the grammatical information to the relevant substructures.

The HPSG grammar itself is viewed as a definition platform for the annotation scheme of the treebank. Hence, the elements of the treebank are not really trees, but feature graphs. Being exemplifications of the phrase-structure backbone, the trees are kept just for reasons of compatibility with other syntactic theories.

As it was mentioned above, the preparation of the annotation scheme requires a proper handling of the following specific tasks:

- Identification of the relevant for Bulgarian part of the HPSG sort hierarchy as described in (Pollard and Sag, 1994); its modification with respect to the language-specific phenomena. It should be noted that the Bulgarian-specific part of the sort hierarchy will be subject to change during the development of the treebank according to the demands of the Bulgarian data.

- Representation of the HPSG Universal Grammar principles from (Pollard and Sag, 1994). This very general grammar is being used as a top constraint during the annotation of the trees in the treebank. For example, each headed phrase in the treebank has to satisfy the Head Feature Principle of HPSG. This grammar is being further extended by Bulgarian specific principles. These specific principles handle, for instance, the relatively free word order in Bulgarian, the clitic behaviour and distribution etc.

- Adapting the lexical and grammatical information from the morphological dictionary of Bulgarian (Popov, Simov and Vidinska, 1998) to the sort hierarchy and to the HPSG principles. This is done during the definition of a tagset to be used within the project.

Hence, the underlying annotation scheme is based on the appropriate language-specific version of the HPSG sort hierarchy. On one hand, such an annotation scheme is very detailed and flexible with respect to the linguistic knowledge, encoded in it. But, on the other hand, because of the massive overgeneration, it is not considered to be annotator-friendly. Thus, the main problem is: how to keep the consistency of the annotation scheme and at the same time to minimise the human work during the annotation. Recall that in our annotation architecture we rely on the cooperative interface between the shallow parsing techniques and HPSG grammar for reducing the unwelcome spurious analyses.

The overall annotation process includes the following steps:

Partial parsing step:

1. Sentence extraction from the corpus. The source of the sentence extraction is the BulTreeBank text corpus. As supporting modules, the CLaRK concordancer and CLaRK grammar engine are relied upon. The concordance software has an access to the morphological information of the words and their syntactic labels, if the relevant grammar had been applied. Thus it allows for queries like: "Find all sentences in which a conjunction is preceded by a verb and followed by an NP."

2. Automatic pre-processing. Each sentence needs first to be pre-processed on all the levels, that precede deeper syntactic annotation. The efficiency of these pre-processing components influences the robustness of next steps. For this reason, we tried to create a reliable pre-processing system, which includes the following modules:

   (i) Morphosyntactic tagging: each word is marked up with the appropriate morphological information, namely: part of speech, gender, number, tense, person, transitivity, etc.;

(ii) Named entity recognition: several named-entity subgrammars have been already developed. These subgrammars follow the ideas from the information extraction and handle not only the closed sets of expressions like numerical expressions and dates, but also open class names like "the city of Sofia" or "Prof. Petrov" and abbreviations. Each of these subgrammars assigns an appropriate description to the signs and thus the processing on the next level, i.e. chunk level, is facilitated.

(iii) Part-of-speech disambiguator: for each ambiguous word the most probable part-of-speech is predicted (see (Simov and Osenova, 2001));

(iv) Partial parsing: the minimal constituents are identified;

We aim at a result of a 100 % accurate partial parsed sentence. The accuracy is checked and validated by a human annotator with the assistance of the appropriate software modules.

HPSG step:

The result from the previous step is encoded into an HPSG compatible representation with respect to the HPSG sort hierarchy. Then it is sent to an HPSG grammar tool, which takes the partial sentence analyses as an input and evaluates all the attachment possibilities for them. The output is encoded as feature graphs.

Resolution step:

Here the output feature graphs from the previous step are further processed in the following way:

1. their intersection is calculated. The intersection exists because all analyses include the partial parsing from the first step and the HPSG grammar tool can not delete information from it;

2. then, on the basis of the differences, a set of constraints over the intersection is introduced;

3. during the actual annotation step, the annotator's task is to extend the intersection to full analysis by adding the missing information. The constraints determine the appropriate extensions and also propagate the information, added by the annotator, in order to minimise the number of the incoming possibilities. To give a simple example, if the annotator has had to specify one of two daughters as the head daughter, the system will automatically percolate the relevant head-features to the mother node and further up the syntactic structure.

This architecture is being currently implemented by establishing an interface between two systems: CLaRK system for XML based corpora development (see (Simov et. al., 2001) and next section) and TRALE system for HPSG grammar development (see (Götz and Meurers, 1997)).

We will use the two systems as follows:

- CLaRK system for:
  - Language resource creation, management and exploration
  - Minimisation of human work during the annotation steps
  - Facilities for semantic validation of the information in language resources (content checking in addition to structure checking)
- TRALE system for:
  - Representation of the HPSG Grammar
  - Logical inference

Obviously, the precision of the information added with each step, decreases. We can be quite certain about the accuracy of the analyses on the morphological level, but the syntactic analyses are incorrect in a large number of cases or contain a high degree of ambiguity. Therefore, each consecutive step requires increased human intervention, with most of it needed on the syntactic level. The aim of the preprocessing is to minimize the amount of this human effort.

To avoid the possible combinatorial explosion on the last level, we plan to apply special compilation techniques which will distribute syntactic information locally onto the overall structure and will encode part of that information as constraints over the structure.

Annotation Process Example

Here we give an example with an ambiguous sentence in order to highlight better the annotation steps and techniques, described above. Note that the HPSG output is simplified and instead of feature graphs we are using context grammar trees in order to demonstrate the main idea.

Sentence extraction

```
<S>the man saw the boy with the telescope
   in the garden</S>
```

Automatic pre-processing

The chosen sentence is tokenized, then the part of speech is assigned to all words in it and some partial grammar for determination of the non-recursive noun phrases is run. The result is:

```
<S>
 <NP><D>the</D><N>man</N></NP>
 <V>saw</V>
 <NP><D>the</D><N>boy</N></NP>
 <P>with</P>
 <NP><D>the</D><N>telescope</N></NP>
 <P>in</P>
 <NP><D>the</D><N>garden</N></NP>
</S>
```

HPSG grammar processing.

The set of all possible syntactic analyses of the partially parsed sentence are returned by the HPSG grammar. This is a set of feature graphs. Each graph in the set is converted to an XML representation. Let us suppose for our example that the grammar returns only three analyses:

Analysis 1:

```
<S>
 <NP><D>the</D><N>man</N></NP>
 <VP>
  <VP>
   <VP>
     <V>saw</V>
     <NP><D>the</D><N>boy</N></NP>
   </VP>
   <PP>
     <P>with</P>
     <NP>
       <D>the</D><N>telescope</N>
     </NP>
   </PP>
  </VP>
  <PP>
    <P>in</P>
    <NP><D>the</D><N>garden</N></NP>
  </PP>
 </VP>
</S>
```

Analysis 2:

```
<S>
 <NP><D>the</D><N>man</N></NP> <VP>
 <VP>
   <V>saw</V>
   <NP>
     <NP><D>the</D><N>boy</N></NP>
     <PP>
       <P>with</P>
       <NP>
         <D>the</D><N>telescope</N>
       </NP>
     </PP>
   </NP>
 </VP>
 <PP>
    <P>in</P>
    <NP><D>the</D><N>garden</N></NP>
 </PP>
 </VP>
</S>
```

Analysis 3:

```
<S>
 <NP><D>the</D><N>man</N></NP>
 <VP>
  <VP>
    <V>saw</V>
    <NP><D>the</D><N>boy</N></NP>
  </VP>
  <PP>
    <P>with</P>
    <NP>
     <NP>
       <D>the</D><N>telescope</N>
     </NP>
     <PP>
        <P>in</P>
        <NP>
          <D>the</D><N>garden</N>
        </NP>
     </PP>
    </NP>
  </PP>
 </VP>
</S>
```

Manual annotation

The intersection of all possible analyses is calculated and represented as an XML document:

```
<S> <NP><D>the</D><N>man</N></NP>
    <V>saw</V>
    <NP><D>the</D><N>boy</N></NP>
    <P>with</P>
    <NP><D>the</D><N>telescope</N></NP>
    <PP>
      <P>in</P>
      <NP><D>the</D><N>garden</N></NP>
    </PP>
</S>
```

Note that the intersection of the three analyses contains more syntactic information than the partial analysis. This is so, because the PP 'in the garden' is shared by all the three analyses. The information from the possible analyses, which is not included into the intersection, is stored as constraints over this document. Then the annotator is asked to complete the intersection to a full analysis. For example, if we mark 'the telescope in the garden' as a single NP, then exactly one analysis is determined and it is not necessary to add more information. Note that in this simple example all the analyses are visible, but in a real case there could be thousands of analyses and their inspection one by one would not be feasible for the annotator without a software support.

4. Software Support - the CLaRK System

In this section we describe the main facilities of the CLaRK system which is the main software used within the BulTreeBank project. CLaRK is an XML-based software system for corpora development. It incorporates several technologies:

- XML technology;
- Unicode;
- Regular Grammars;
- Constraints over XML Documents.

4.1. XML Technology

At the heart of the CLaRK system is the XML technology implemented as a set of utilities for structuring, manipulation and management of data. We chose the XML technology because of its popularity, its ease of understanding and its already wide use in description of linguistic information. Besides the XML language processor itself, we implemented an XPath language engine for navigation in documents and an XSLT language engine for transformation of XML documents. We started with basic facilities for creation, editing, storing and querying of XML documents and

developed further this inventory towards a powerful system for processing not only single XML documents but an integrated set of documents and constraints over them. The main goal of this development is to allow the user to add the desirable semantics to the XML documents.

The core of CLaRK is an XML editor, which is the main interface to the system. Each loaded into the editor document is presented to the user in two or more views. One of these views reflects the tree structure of the document. The other views of the document are textual. Each textual view shows the tags and the text content of the document. The tags in the textual view are separated from the rest of the text and can not be edited. The user has the possibility of attaching to each textual view a filter, which determines the tags and the content of the elements to be displayed in the view. This option allows hiding some of the information in the document and concentrating on the rest of it. To the different textual views of the same document the user can attach different filters. The editor supports a full set of editing operations as copy, cut, paste and so on. These operations are consistent with the XML structure of the document. Thus the user can copy or delete a whole subtree. Some of these editing operations as search and replace are defined in terms of XPath expressions. Thus the user can search not only in the textual content of the document but also with respect to the XML mark-up.

4.2. Tokenization

XML considers the content of each text element as a whole string that is unacceptable for corpus processing. For this reason it is required for the wordforms, punctuation and other tokens in the text to be distinguished. In order to cope with this problem, the CLaRK system supports a user-defined hierarchy of tokenizers. At the very basic level the user can define a tokenizer in terms of a set of token types. In this basic tokenizer each token type is defined by a set of UNICODE symbols. Above this basic level tokenizers the user can define other tokenizers for which the token types are defined as regular expressions over the tokens of some other tokenizer, the so called parent tokenizer. For each tokenizer an alphabetical order over the token types is defined. This order is used for operations like the comparison between two tokens, sorting and similar.

Sometimes in different parts of one and the same document the user might need to apply different tokenizers. For instance, in a multilingual corpus the sentences in different languages have to be tokenized by different tokenizers. In order to allow this flexibility, the system allows for attaching tokenizers to the documents via the DTD of the document. To each DTD the user can attach a tokenizer which will be used for tokenization of all textual elements of the documents corresponding to the DTD. Additionally, the user can overwrite the DTD tokenizer for some of the elements, attaching to them other tokenizers.

4.3. Regular Grammars

The basic CLaRK mechanism for linguistic processing of text corpora is the regular grammar processor. The application of the regular grammars to XML documents is connected with the following problems:

- how to treat the XML document as an input word for a regular grammar;
- how the return-up grammar category to be incorporated into the XML document; and
- what kind of 'letters' to be used in the regular expressions so that they correspond to the 'letters' in the XML document.

The solutions to these problems are described in the next paragraphs.

First of all, we accept that each grammar works on the content of an element in an XML document. The content of each XML element (excluding the EMPTY elements on which regular grammar cannot be applied) is either a sequence of XML elements, or text, or both. The input word for a grammar is calculated in the following way: each textual element of the content is tokenized by an appropriate tokenizer and each XML element is substituted by a list of values returned by an XPath expression and evaluated over the node. In order to recognize whether a value in the input word is from a text or from an element, the CLaRK system requires that the values of an XML element were enclosed in angle brackets: `<...>`. For instance, the content of the following element:

```
<s>John <v>loves</v> Mary</s>
```

forms the following input word under appropriate tokenizers and XPath values:

```
"John" " " < "v" > " " "Mary"
```

In order to map the values in the input word, we use token descriptions in the regular expressions. The simplest token descriptions are the tokens themselves. Additionally, in the token description we can use wildcard symbols # for zero or more symbols, @ for zero or one symbol, and token categories. Each token description in a regular expression is matched exactly to one token in the input word. It is important to see that the value of an XML element is not just the tag of the element. For example, if we have the following element:

```
<s>
 <w g="N">John</w>
 <w g="V">loves</w>
 <w g="N">Mary</w>
</s>
```

then the sequence of tags is simply `<"w"> <"w"> ...`, which is not intended to be an input word for a grammar. In order to get the values of the attributes of the elements, we have to use the following XPath expression: attribute::g. And then the input word will be: `<"N"> <"V"> <"N">`. Writing complex XPath expressions, the user can determine very complicated values for the XML elements depending of the context of their use.

The last problem when applying grammars over XML documents is how to incorporate the category assigned to a given rule. In general we accept that the category has to be encoded as an XML mark-up of arbitrary complexity. For instance, the following simple tagger (example is based on (Abney, 1996)):

```
"the"|"a" -> Det
"telescope"|"garden"|"boy" -> N
"slow"|"quick"|"lazy" -> Adj
"walks"|"see"|"sees"|"saw" -> V
"above"|"with"|"in" -> Prep
```

can be encoded in CLaRK system as:

```
"the"|"a" -> <Det>\w</Det>
"telescope"|"garden"|"boy" -> <N>\w</N>
"slow"|"quick"|"lazy" -> <Adj>\w</Adj>
"walks"|"see"|"sees"|"saw" -> <V>\w</V>
"above"|"with"|"in" -> <Prep>\w</Prep>
```

where \w is a variable for the recognized word. The mark-up defining the category can be as complicated as necessary. The variable \w can be repeated as many times as necessary (it can also be omitted).

4.4. Constraints over XML documents

Several mechanisms for imposing constraints over XML documents are available. The constraints cannot be stated by the standard XML technology. The following types of constraints are implemented in CLaRK: 1) finite-state constraints - additional constraints over the content of given elements based on a document context; 2) number restriction constraints - cardinality constraints over the content of a document; 3) value constraints - restriction of the possible content or parent of an element in a document based on a context. The constraints are used in two modes: checking the validity of a document regarding a set of constraints; supporting the linguist in his/her work during the building of a corpus. The first mode allows the creation of constraints for the validation of a corpus according to given requirements. The second mode helps the underlying strategy of minimisation of the human labour.

General syntax of the constraints in the CLaRK system is the following:

```
(Selector, Condition, Event, Action)
```

where the selector defines in which node(s) in the document the constraint is applicable; the condition defines the state of the document when the constraint is applied. The condition is stated as an XPath expression which is evaluated with respect to each node selected by the selector. If the evaluation of the condition is a non-empty list of nodes then the constraints are applied; the event defines some conditions of the system when this constraint is checked for application. Such events can be: the selection of a menu item, the pressing of key shortcut, some editing command as enter a child or a parent and similar; the action defines the way of the actual application of the constraint.

Here we present constraints of type ”Some Children”. This kind of constraints deal with the content of some elements. They determine the existence of certain values within the content of these elements. A value can be a token or an XML mark-up and the actual value for an element can be determined by the context. Thus a constraint of this kind works in the following way: first it determines to which elements in the document it is applicable, then for each such element in turn it determines which values are allowed and checks whether in the content of the element some of these values are presented as a token or an XML mark-up. If there is such a value, then the constraint chooses next element. If there is no such a value, then the constraint offers to the user a possibility to choose one of the allowed values for this element and the selected value is added to the content as a first child. Additionally, there is a mechanism for filtering of the appropriate values on the basis of the context of the element.

This kind of constraints is very useful for filling of the content of elements with predetermined set of values as, for example, in dictionary construction. Within BulTreeBank, we use these constraints for manual disambiguation of morpho-syntactic tags of wordforms in the text. For each wordform we encode the appropriate morpho-syntactic information from the dictionary as two elements: <aa> element which contains a list of morpho-syntactic tags for the wordform separated by a semicolon, and <ta> element which contains the actual morpho-syntactic tag for this use of the wordform. Obviously the value of <ta> element has to be among the values in the list presented in the element <aa> for the same wordform. ”Some Children” constraints are very appropriate in this case. Using different conditions and filters on the values we implemented and used more than 50 constraints during the manual disambiguation of wordforms in the “golden standard” of the project. It is important to mention that when the context determines only one possible value for some word, it is added automatically to the content of <ta> element and thus the constraint becomes a rule.

4.5. Additional Facilities

Besides the basic tools presented so far, there is a great number of additional facilities for supporting corpus development implemented in the CLaRK system. The Extractor is a tool for extracting elements from XML documents in a new document. For instance this can be used to extract all sentences in certain documents that meet some condition. The condition is stated as an XPath expression. The Remove tool allows the user to delete some elements that meet a certain condition stated again as an XPath expression. The Sorting tool is concerned with sorting elements of a document according to some keys defined over these elements. The sorting is defined in terms of two XPath expressions. The first expression determines which elements will be sorted. This expression is evaluated with respect to the root of the document as a context node. The second XPath expression defines the key for each element and it is evaluated for each node returned by the first XPath expression. The list of nodes returned by the first expression is sorted according to the keys of the nodes. Afterwards the nodes are returned in the document in the new order.

A concordance tool is implemented on the basis of the XPath engine, regular grammar engine and a sorting module. The first step in a concordance construction is to extract the relevant information from some documents. This is done by an XPath expression which is evaluated and the returned list of nodes is stored as a separate document. The extracted elements are ordered in an appropriate way. For example, in case of appropriately marked-up corpus one

can extract all verbs and order them with respect to the first noun on the right hand side of the verb (not necessarily the first word on the right hand side).

4.6. Cascade Processing

The central view of the use of the CLaRK system is that an XML document under processing can be seen as a "blackboard" on which different tools can write some information, reorder it or delete it. The user can order the applications of the different tools so as to achieve the necessary processing. We call this possibility **cascade processing** after the cascade regular grammars. In this case we can order not just different grammars but also other tools like constraints, removal operations, transformations and sorting. In this section we present two uses of this paradigm within the BulTreeBank project.

The first one is concerned with sentence boundary delimitation. This is a well known problem due to the fact that some abbreviations end with a full stop which can also be end of a sentence. We solve this problem in the following way. First we run a grammar which recognizes the abbreviations in the text and annotates them. The annotation contains information about the abbreviation full stop, but it cannot be deleted from the token list because the same full stop can also be the end of a sentence. In order to distinguish such full stops, the grammar for the abbreviations leaves the full stops in the token lists, but it changes their mark-up. For example, if we have in the text the Bulgarian abbreviation `<w>str</w> <pt>.</pt>`, after running the grammar we will have `<abbr><w>str</w> <pt>.</pt></abbr> <pt type="abbr">.</pt>`. On the next step of processing, the grammar for sentence boundary delimitation will consider also the full stops marked-up with the above `type` attribute. If the grammar recognizes some of these full stops as the end of sentences, it will replace their mark-up with mark-up for the end of the sentence. After the grammar for sentence delimitation is applied, we run a removal operation to delete all full stops that are part of an abbreviation, but not end of a sentence.

The second use of this approach deals with the fact that some sequences of words can be automatically disambiguated. Sometimes such sequences of words cannot be constituents, and their eventual grouping is not part of the analysis of the sentences. In this case we proceed in following way. First we write a grammar that groups together such words and marks them up in an appropriate way. Then we run a sequence of constraints that disambiguates the words within such a group. After the constraints, we run a removal operation to delete the group mark-up. An example of such a case is the so called 'da'-construction in Bulgarian. It includes the conjunction 'da', a number of clitics and a verb form. Within such groups, 'da' is ambiguous between a conjunction and a particle, but it is defined as a conjunction. Most of the clitics are ambiguous between personal pronouns and possesive pronouns, but in this context they can be personal pronouns only. The verb can only be in the present tense. Using such a mechanism, we succeeded to reduce the number of ambiguities with more than 10%.

5. Acknowledgements

The work reported here is done within the BulTreeBank project. The project is funded by the Volkswagen Stiftung, Federal Republic of Germany under the Programme *"Cooperation with Natural and Engineering Scientists in Central and Eastern Europe"* contract I/76 887.

6. References

Steve Abney. 1991. *Parsing By Chunks.* In: *Robert Berwick, Steven Abney and Carol Tenny (eds.), Principle-Based Parsing.* Kluwer Academic Publishers, Dordrecht.

Steve Abney. 1996. *Partial Parsing via Finite-State Cascades.* In: *Proceedings of the ESSLLI'96 Robust Parsing Workshop.* Prague, Czech Republic.

Dipper, S. 2000. *Grammar-based Corpus Annotation*. In *Proceedings of the Workshop on Linguistically Interpreted Corpora*, Luxembourg, August 6

Thilo Götz and W. Detmar Meurers. 1997. *The ConTroll system as large grammar development platform*. In *Proceedings of the ACL/EACL post-conference workshop on Computational Environments for Grammar Development and Linguistic Engineering*. Madrid, Spain.

Paul J. King. 1989. *A Logical Formalism for Head-Driven Phrase Structure Grammar*. Doctoral thesis, Manchester University, Manchester, England.

Paul J. King and Kiril Iv. Simov. 1998. The automatic deduction of classificatory systems from linguistic theories. In *Grammars*, volume 1, number 2, pages 103-153. Kluwer Academic Publishers, The Netherlands.

Carl J. Pollard and Ivan A. Sag. 1987. *Information-Based Syntax and Semantics*, vol. 1. CSLI Lecture Notes 13. CSLI, Stanford, California, USA.

Carl J. Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, Illinois, USA.

D. Popov, K. Simov, and S. Vidinska. 1998. *A Dictionary of Writing, Pronunciation and Punctuation of Bulgarian Language*, Atlantis SD, Sofia, Bulgaria.

Kiril Simov. 2001. *Grammar Extraction from an HPSG Corpus.* In: Proc. of the RANLP 2001 Conference, Tzigov chark, Bulgaria, 5-7 Sept., pp. 285-287.

Kiril Simov, Zdravko Peev, Milen Kouylekov, Alexander Simov, Marin Dimitrov, Atanas Kiryakov. 2001. *CLaRK - an XML-based System for Corpora Development.* In: Proc. of the Corpus Linguistics 2001 Conference, pages: 558-560.

Kiril Simov, and Petya Osenova. 2001. *A Hybrid System for MorphoSyntactic Disambiguation in Bulgarian.* In: Proc. of the RANLP 2001 Conference, Tzigov chark, Bulgaria, 5-7 Sept., pp. 288-290.

K. Simov, G. Popova, and P. Osenova. 2001. *HPSG-based syntactic treebank of Bulgarian (BulTreeBank).* In: *"A Rainbow of Corpora: Corpus Linguistics and the Languages of the World"*, edited by Andrew Wilson, Paul Rayson, and Tony McEnery; Lincom-Europa, Munich, pp. 135-142.