Extended Dependency Unification Grammar

Peter Hellwig, University of Heidelberg

1 A Sketch of Dependency Unification Grammar

A quick way of gaining a first impression of a formalism is looking at concrete examples.

1.1 Representation of Natural Language Input

The sentence

Arthur attends the Prague meeting.

is turned into the following representation by the PLAIN parser:

```
(UTTERANCE: assertion': clause type[main]
    (< PROPOSITION: attend: verb form[finite] tense[present] person[3rd]
    number[singular]
        (< SUBJECT: Arthur: noun person[3rd] number[singular])
        (> DIROBJECT: meeting: noun
            (< DETERMINATION: definit': determiner sequence[1])
            (< ATTRIBUTE: Prague: noun sequence[2]))));
```

Fig. 1: Representation of a sentence

Fig. 1 is an instance of the so-called Dependency Representation Language (DRL). DRL is used for all purposes in the PLAIN system, e.g. for representing parsing output, for storing information in a knowledge base, for writing grammars and lexica, for writing semantic rules, and as the source for generating natural language output. DRL is, as it were, the programming language of the linguist in order to make the system analyse and process a particular natural language.

The expressions of DRL are lists (in the technical sence of list processing) consisting of terms; each sublist is included in parentheses, including sublists consisting of just one term. A list corresponds to a graph, in particular, to a tree. The structure of the tree in fig. 1 is elucidated by the indenting of the terms.

Each term carries three kinds of labels: a role, a lexeme, and a complex category. The latter consists of a main category and a set of grammatical attributes followed by their values in square brackets. The symbols "<" and ">" indicate that the subsequent subtree corresponds to a segment in the input which is left or right to the segment corresponding to the term dominating the subtree. (The top-term in fig. 1 corresponds to the period; that is why the subtree headed by the verb is marked to be on the left of the term with the "assertion" label.) In addition, positional features can be included in the set of grammatical attributes, e.g. the attribute "sequence" which regulates the sequence of the determiner *the* and the noun *Prague*.

A peculiarity of DUG, which will be discussed in detail below, is the fact that the role of a term refers to the whole subtree it dominates, e.g. UTTERANCE refers to the whole sentence, PROPOSITION includes everything dominated by the verb, DIROBJECT is the role of the subtree corresponding to the noun phrase *the Prague meeting* etc. The rest of the information in fig. 1 should be self-explaining.

1.2 Grammatical Description

The shape of a grammar can be very different from the underlying syntactic representation. For example, the structural representation might consist of phrase structure trees while the grammar either consists of a set of rewriting rules, an augmented transition network, or a lexicon with complex entries as it is the case in categorial grammar. A DUG description is identical to the structural representation assigned to an utterance, except for taking apart the trees into single head-to-dependent term relationships and for introducing variables for paradigmatically substitutable entities, i.e. a DUG consists of a set of templates which directly mirror potential dependency trees. The individual templates necessary to yield the tree in fig. 1 are the following:

```
1   (UTTERANCE: assertion': clause type[main]
        (< PROPOSITION: = : verb form[finite]));

2   (*: attend : verb form[finite]
        (< SUBJECT: = : noun person[C] number[C]));

3   (*: attend : verb
        (> DIROBJECT: = : noun));

4   (*: meeting: noun
        (< DETERMINATION: = : determiner sequence[1]));

5   (*: meeting: noun
        (< ATTRIBUTE: _ : noun sequence[2]));
```

Fig. 2: Individual templates

Templates can be seen as an application of the frame concept suggested by Marvin Minsky. The words associated with the lexemes in the head terms open up slots for obligatory or potential complements.

The role of each complement is predictable and, hence, marked by a role label. The lexical filling of each slot is arbitrary and, therefore, substituted by the variables "=" (obligatory) and "_" (potential). The morpho-syntactic features of the potential fillers of slots are restricted, as far as necessary, by means of categories.

The templates should be as general as possible. The heads often carry a role variable ("*"), since they, in turn, might fit in various slots. Grammatical features which are not distinctive for the given template are simply omitted. (Compare the individual categories in fig. 1 with the more general ones in fig. 2.) The mechanism of fitting fillers into slots adheres to the common subsumption principles of unification grammars. The agreement between heads and dependents is taken care of by the special value C, e.g. "person[C]" and "number[C]" in the subject slot. The values of an attribute which is marked by "C" in a dependent term are unified with the values of the same attribute in the dominating term. If, for example, a filler of the subject slot in template 2 has "person[1st]" then the verb must also have the feature "person[1st]". The C-value is a means to make templates more general and, in addition, to propagate feature values through the whole tree.

All of the templates in fig. 2 are dominated by an individual lexeme. Similar to a categorial grammar, a DUG can be considered to consist of a lexicon only. This lexical approach to syntax will be further discussed below. The only syntactic rule in a DUG is the instruction to fit fillers into slots.

It is a fact, however, that many words open up similar slots. In order to capture this regularity, generalized templates are set up which describe types of complements. A particular type of complement is associated with a lexical item via a so-called reference. The identification of generalized templates is achieved by a name which is stored in the lexeme position of the dominating term. The set of templates in fig. 2 can now be replaced by the following sets of templates and references:

```
1   (UTTERANCE: +proposition: clause type[main]
        (< PROPOSITION: = : verb form[finite]));

2   (*: +subject: verb form[finite]
        (< SUBJECT: = : noun person[C] number[C]));

3   (*: +dirobject: verb
        (> DIROBJECT: = : noun));

4   (*: +determination: noun
        (< DETERMINATION: = : determiner sequence[1]));

5   (*: +attribute: noun
        (< ATTRIBUTE: _ : noun sequence[2]));
```

Fig. 3: Generalized templates

```
1   (: ->  (*: assertion')
           (*: +proposition));

2   (: ->  (*: attend)
           (& (*: +subject) (*: +dirobject)));

3   (: ->  (*: meeting)
           (& (*: +determination) (*: +attribute)));
```

Fig. 4: References

The system looks up the lexeme of a word first (see fig. 5 below), then it looks up the reference for this lexeme in fig. 4, then it looks up the templates in fig. 3 mentioned in the reference and, finally, it subordinates the slots in the templates to the lexeme in question. The outcome of this procedure are exactly the individual templates shown in fig. 2.

With respect to content, the generalized templates correspond to the derivation rules of a generative grammar (in the framework of a dependency structure, though) and the references correspond to the subcategorization. The word orientation is preserved, however. What we have is, so to speak, a lexicon (the references) with a formal explication of the subcategorization symbols (the templates). This organization fits nicely with dictionaries for human use, e.g. the Longmans Dictionary of Contemporary English.

What is still missing is the morpho-syntactic description of the individual word forms. In principle, this description must map strings of the input language into DRL expressions. A simple morpho-syntactic lexicon for the above example might look as follows:

'Arthur'	(*: Arthur: noun person[3rd] number[singular])*;*
'attends'	(*: attend: verb form[finite] tense[present] person[3rd] number[singular]);
'the'	(*: definit': determiner);
'Prague'	(*: Prague: noun person[3rd] number[singular])*;*
'meeting'	(*: meeting: noun person[3rd] number[singular])*;*
'.'	(UTTERANCE: assertion': clause type[main]*);*

Fig. 5: Morpho-syntactic lexicon

Each input word is replaced by the corresponding DRL-term. This DRL-term is unified with the role and categories of the templates attributed to the lexeme by means of a reference. If the morpho-syntactic categories of the original term are not subsumed by the categories in the template then the template is not applicable. This is the case if, for example, "+subject" is in the reference, but the feature "form[finite]" is not met by the given verb.

The factual morpho-syntactic lexicon of a DUG differs from the one in fig. 5 in that the strings of natural language might be composed of prefixes, varying stems, derivational elements and inflectional endings. Similar to the subcategorization templates, there are generalized morpho-syntactic patterns which yield the same categories for a large amount of different words. However, I do not want to go deeper into morphology here.

1.3 Parsing

The PLAIN parser is a bottom-up chart parser. In principle, the algorithm is very simple. At the end of the lexicon phase, there is (at least) one DRL list for each segment (e.g. word) in the input, containing the complete morpho-syntactic categorization of the segment as well as the slots attributed to the segment via references and templates. These lists form the initial contents of the parsers working space:

```
-----------------------------------------------------------
1-1   'Arthur'

(*: Arthur: noun person[3rd] number[singular]);
-----------------------------------------------------------
2-2   'attends'

(*: attend: verb form[finite] tense[present] person[3rd] number[singular]
      (< SUBJECT: = : noun person[C] number[C])
      (> DIROBJECT: = : noun));
-----------------------------------------------------------
3-3   'the'

(*: definit': determiner);
-----------------------------------------------------------
4-4   'Prague'

(*: Prague: noun person[3rd] number[singular]);
-----------------------------------------------------------
5-5   'meeting'

(*: meeting: noun person[3rd] number[singular]
      (< DETERMINATION: = : determiner sequence[1])
      (< ATTRIBUTE: _ : noun sequence[2]));
-----------------------------------------------------------
6-6   '.'

(UTTERANCE: assertion': clause type[main]);
      (< PROPOSITION: = : verb form[finite]));
-----------------------------------------------------------
```

Fig. 6: Initial contents of the chart

The combination capabilities of the segments are completely transparent. No further resources are needed in order to construct larger segments from the smaller ones. The categories in the slots are simply unified with the categories of the heads of potential fillers. The subsequent entries of the chart should be self-explaining:

```
------------------------------------------------------------
1-2   'Arthur attends'

(*: attend: verb form[finite] tense[present] person[3rd] number[singular]
        (< SUBJECT: Arthur : noun person[3rd,C] number[singular,C])
        (> DIROBJECT: = : noun));
------------------------------------------------------------
4-5   'Prague meeting'

(*: meeting: noun person[3rd] number[singular]
        (< DETERMINATION: = : determiner sequence[1])
        (< ATTRIBUTE: Prague: noun sequence[2]));
------------------------------------------------------------
3-5   'the Prague meeting'

(*: meeting: noun person[3rd] number[singular]
        (< DETERMINATION: definit' : determiner sequence[1])
        (< ATTRIBUTE: Prague: noun sequence[2]));
------------------------------------------------------------
1-5   'Arthur attends the Prague meeting'

(* : attend: verb form[finite] tense[present] person[3rd] number[singular]
        (< SUBJECT: Arthur : noun person[3rd,C] number[singular,C])
        (> DIROBJECT: meeting: noun
                (< DETERMINATION: definit' : determiner sequence[1])
                (< ATTRIBUTE: Prague: noun sequence[2])));
------------------------------------------------------------
1-6   'Arthur attends the Prague meeting.'

(UTTERANCE: assertion': clause type[main]);
        (< PROPOSITION: attend: verb form[finite] tense[present] person[3rd]
        number[singular]
                (< SUBJECT: Arthur : noun person[3rd,C] number[singular,C])
                (> DIROBJECT: meeting: noun
                        (< DETERMINATION: definit' : determiner sequence[1])
                        (< ATTRIBUTE: Prague: noun sequence[2]))));
------------------------------------------------------------
```

Fig. 7: Further contents of the chart

The last entry in the chart covers the whole input, from word 1 up to word 6. The DRL-list in this chart element is the desired output.

1.4 Semantic Processing

There is no space, here, to go deeply into semantics. It is important, however, that there is a concept of semantics at the very beginning of designing a syntax formalism. Otherwise it might be hard to construct an interface between the outcome of the parser and the subsequent components of the NLP system.

I advocate the Wittgenstein view that the meaning of language is its use. In the framework of a computer system, this view can be nicely made concrete: Semantics is anything the system does with the input after analysis. What can be done with DRL-trees is to transform them into one another and, in doing so, simulate operations which humans perform with language. The transformation is guided by production rules written in DRL. Let us, again, illustrate this by an example.

(1) Input: *Arthur attends the meeting.*

(2) Representation:

```
(UTTERANCE: assertion'
        (< PROPOSITION: attend
                (< SUBJECT: Arthur
                (> DIROBJECT: meeting
                        (< DETERMINATION: definit')
                        (< ATTRIBUTE: Prague))));
```

(3) Passive-rule:

```
(: =>
        (PROPOSITION: #
                (< SUBJECT: $1)
                (> DIROBJECT: $2))
        (PROPOSITION: be_passive
                (< SUBJECT: $2)
                (> PREDICATIVE: #
                        (> AGENTIVE: by
                                (> : $1)))));
```

(4) Transformed representation:

```
(UTTERANCE: assertion'
        (< PROPOSITION: be_passive
                (< SUBJECT:  meeting
                        (< DETERMINATION: definit')
                        (< ATTRIBUTE: Prague))
                (> PREDICATIVE: attend
                        (> AGENTIVE: by
                                (> : Arthur)))));
```

(5) Output: *The Prague meeting is attended by Arthur.*

Fig. 8: Producing a passive-paraphrase

The morpho-syntactic categories are removed from the DRL-representation in fig. 8. This is justified because they describe the surface form of natural language. (In fact, they are introduced again between step (4) and (5) in the example, in order to create the right morphological forms for the output. Let us leave this aside, though.) Just by removing the morphy-syntactic information, DRL is turned into a handy knowledge representation for semantic purposes.

Rules, like the passive-rule in the example, consist of a term carrying a rule symbol and two subtrees, both possibly containing variables. The first subtree represents a template for a sublist to which the rule is to be applied. The second subtree describes the sublist to be constructed by the rule. Variables in the first subtree are bound to parts of the input list. A variable beginning with "#" can bind one term; a variable beginning with "$" can bind a term plus the tree that it dominates. The resulting list is identical to the second subtree in the rule, except that variables which were bound before are substituted by the corresponding parts of the input list.

In the example, the rule is applied within the original representation, i.e. the subtree matching the first part of the rule is replaced by the subtree that is produced by the rule. The original representation is changed. Other applications of rules might derive completely new lists from existing ones. This is the case with inference rules. They augment the existing knowledge base rather than replace it.

We have applied this technology to many areas, e.g. to translation, logical deduction and theorem proving. Any semantic relationship between natural language expressions can be formulated in terms of a relationship between DRL-lists. Any such relationship can, then, turned into an operation of the semantic component by writing an appropriate rule.

1.5 Extensions

There are many additional provisions in a DUG to ease the description of syntactic phenomena. For example, it is not only possible to specify potential complements by means of templates, but one can also describe potential heads for a given adjunct. The latter technique is reasonable in the case of adverbials which can go with almost any verb. It would be very inefficient to include all possible adverbial in the valency description of each verb.

A principal extension of the DUG formalism is required in order to cover coordination. Coordination is a linear phenomenon which overlays the hierarchical structure of governors and complements. Compare the following sentence:

Arthur attends the Prague meeting and Sheila the Paris workshop.

The verb *attend* allows for two complements, a subject and an object. However, these complements are occupied twice in the sentence and connected via the conjunction *and*. How can this fact be represented in a dependency tree?

It has been suggested to consider coordination as an operation on syntactic structures rather than as a syntactic construction on its own (see Henning Lobin, forthcoming). This means, however, that the concept of a purely declarative grammar guiding the parser is at stake. It is of little help to point out that the sentence above is derived from the two sentences *Arthur attends the Prague meeting* and *Sheila attends the Paris workshop*. Due to the delimitation of syntax and semantics in DUG, there must be a representation of the coordinated structure first which, subsequently, might be transformed into two sentences by the semantic component.

The following dependency tree solves the representation problem rised by the example sentence:

```
(UTTERANCE: assertion': clause type[main]
    (< PROPOSITION: attend: verb form[finite] tense[present] person[3rd]
    number[singular]
        (< SUBJECT: Arthur: noun person[3rd] number[singular])
        (> DIROBJECT: meeting: noun
            (< DETERMINATION: definit': determiner sequence[1])
            (< ATTRIBUTE: Prague: noun sequence[2]))
        (> CONJUNCT: and
            (> SUBJECT: Sheila: noun person[3rd] number[singular])
            (> DIROBJECT: workshop: noun
                (< DETERMINATION: definit': determiner sequence[1])
         (> ATTRIBUTE: Paris: noun sequence[2])))));
```

Fig. 9: Representation of a sentence with coordination

The conjunction *and* dominates the complements which are in its scope, namely *Sheila* and *the Paris workshop*. The conjunction is a "pseudo-head", though, i.e. the roles and categories keep to be determined by the original head *attend*. The conjunction is transparent, as it were, with respect to the roles and the morpho-syntactic categories of the dependents. *Sheila* is SUBJECT of *attend* as well as *Arthur*, and *the Paris workhop* is DIROBJECT of *attend* as well as *the Prague meeting*. It is not difficult to write rules which enable the semantic component to extract from fig. 9 the two simple sentences which underly the coordination.

The next task is to introduce special types of templates by means of which the grammar writer can describe conjunctions and their syntactic behavior. Several types of conjunctions must be distinguished, e.g. infix conjunctions like *and* and *or* as well as prefix conjunctions like *either ... or.* Ellipsis has to be taken care of in the descriptions, too. The formal apparatus invented for this purpose cannot presented here, due to lack of space. It seems, however, that DUG and the PLAIN parser are apt to handle the vast majority of coordinations in German.

2 Points of Discussion

2.1 The Word-Based Approach to Syntax

For centuries, it has been the traditional viewpoint that syntax deals with the combination capabilities of words. This syntax paradigm changed drastically when Noam Chomsky defined a language as a set of sentences and provided the formal means to (theoretically) generate this set. Since then, syntax is usually conceived as the abstract structure of non-terminal units, in particular, sentences; words are just filled into the slots of such abstract entities.

Although the level of formal clarity of generative grammar is obliging, there are good reasons to consider the old view to be more adequate. Words are not just fillers of a-priori structural patterns; they create these patterns. Some words in a sentence semantically denote relations, others denote the entities standing in these relations; some words are operators, others are in the scope of these operators. There is usually an asymmetry between immediate constituents. One constituent is the governor, it gives rise to the whole construction and determines it. The other constituents are the dependents, i.e. they adjust to the demands imposed on the construction by the governor.

These facts are known and widely accepted. Nevertheless, sentence-based formalisms are still prevailing. The abstract symbol S is made the starting point; the governing properties of words somehow intervene with the derivation rules by tricky mechanisms. If the relationship between sentence and words is literally turned upside-down, starting with the combination capabilities of words, the representations become simpler. Side effects, due to mathematical complexity, are less likely to occur when new data is included.

The lexicalization of grammatical phenomena is a prerequisite for broad coverage applications. A lexicon is a special type of knowledge representation characterized by modularity and independent accessibility of the information. This type of data representation is used in any large and complex domain, not only in linguistics. The ease of lexicalization depends on the structure imposed on the object domain. Phrase structures are comparitively hard to lexicalize, a reason why categorial grammars (without additional rules) have not been widely used. Word-based dependency structures, on the contrary, can be easily divided into units which can be listed in a lexicon.

2.2 Constituency and Dependency

I assume that a constituent is any unit, small or large, to be distingished in an utterance. Dependency grammars have been less appealing because they seem to abandon the notion of constituents. Each node in a dependency tree represents a word. Therefore, the dependency relation has often been misconceived as a word-to-word relationsship. However, the presumptive dispensation of non- terminal symbols does not imply the disposal of constituents. I hope that DUG convinces potential converts from phrase structure grammars that they will loose nothing.

In my opinion, the essential concept of a dependency grammar is the complement; a dependency relation is a word-to-complement relationship. Obviously, a complement can consist of many words. The object complement of the verb *attend* in our example sentence above is, of course, the phrase *the Prague meeting* rather than the word *meeting*. On the other hand, the subject complement of *attend* could have as well been the phrase *all friends of Petr Sgall* instead of the single word *Arthur*.

Internally, a complement is structured according to word-to-complement principles, too, and is represented as such. Therefore, all nodes in a dependency tree eventually correspond to words (roughly speaking - leaving aside a potential decomposition of words into smaller units). Nevertheless, each complement relationship holds between the word of a dominating node and the complement represented by a whole subtree rather than between individual words or nodes.

Templates, like the ones in fig. 3, are to be filled by subtrees, i.e. they decribe phrases and not just single words. For example, the "+subject"-template covers *Arthur, you*, *all the friends of Petr Sgall* as well as any other subject. Assigning this template to a verb means the same as saying that a subject NP can go with the verb.

It is obvious that a DUG accounts for non-terminal constituents. The roles are clearly non-terminal symbols, since they denote complements (see next section). The morpho-syntactic category in a head term might have undergone changes by upward propagation of features from its dependents; so it is virtually non-terminal too. Nevertheless, the part of speech of the head term is usually an appropriate category for the whole complement. This fact supports the view that it is the word that determines a syntactic construction rather than an abstract pattern.

Constituency and dependency are by no means antagonisms. What is true is that the word-to-complement principle of a dependency grammar imposes a special constituency structure on a sentence. In each construction, one constituent is a lexical item (the governor) the other one is a word, a phrase or a clause (the complement).

The parser profits from the word-and-complement constituency. Common phrase structure parsers either proceed top-down or bottom-up; the top-down direction is said to be expectation-driven, the bottom-up direction to be data-driven. It is known that both approaches have advantages and disadvantages and that a combination of both principles would be ideal. The DUG parser is data-driven, since it starts with the words in the input and proceeds bottom up. At the same time, it is expectation-driven, since the templates associated with each word introduce a top-down view. In fact, we have data-driven expectations.

2.3 Syntactic Roles

A template can be filled by a subtree of more or less internal complexity and, hence, by a smaller or larger phrase. Moreover, certain phrases of rather different form are felt to constitute the same complement:

That Arthur attends the meeting amazes him.

Your behavior amazes him.

What you did amazes him.

The fact that there is a common syntagmatic role which is varied in these examples becomes obvious when we ask:

What amazes him?

The answers are:

That Arthur attends the meeting.

Your behavior.

What you did.

A separate set of symbols must be available to denote the functional identity of complements, while the morpho-syntactic categorization marks the differences. The role labels in DUG were introduced for this purpose.

Originally I was inspired by Charles Fillmore's Deep Cases, feeling uneasy, however, about the lack of criteria for assigning roles like Agent, Instrument, Patient etc. What is the role of a given syntagmatic unit? In the concrete case, one can always give a concrete answer to this question. For example, the role of the phrase *that Arthur attends the meeting* in the first sentence above is obviously to denote the amazing fact. The same is true for *your behaviour* and *what you did* in the second and third sentence, while the role of *him* in all the examples can be characterized as the denotation of the one being amazed.

I call this test "the lexical elucidation of syntactic roles". The test is a sufficient criterion for distinguishing and correlating complements. If you get the same answer you have the same role; if you get a different answer you have a different role. Of course, the answer depends on the lexical item which is the governor of the complement. Fortunately, this lexical item is always at hand in a grammatical representation which is based on the word-to-complement relationship. In fact, it would be redundant to symbolize the concrete roles in the tree, since the governing lexeme, which determines the roles, is always present. The only requirement, at the moment, are any distinctive labels for the complements under the same governing term.

It is the task of the semantic component to correlate the various complement frames of a word, e.g. in *Hiram sprayed the wall with paint* and *Hiram sprayed the paint on the wall* (see Stanley Starosta's paper). Since governing lexemes continue to be present in the representations, the role labels in the rules still have to be identical or distinctive with respect to the concrete governor only. Nevertheless, it is to be expected that the attempt to write semantic rules will gradually reveal a more abstract notion of roles. A first hint is rendered by coordination. On the one hand, we know that the same role is a prerequisite for the coordination of complements; that is why * *Hiram sprayed the wall and the paint* is ill-formed. On the other hand, ellipsis in coordination shows that a complement can belong to different lexical items at the same time. Hence, its role must be the same under both heads, e.g. *Hiram attended the Prague meeting and sprayed the wall with paint.* I suspect that the set of emerging functional categories will include roles which grammarians have distinguished intuitively since long, namely subject, direct object, indirect object, attribut, etc., although the final set will eventually be finer-grained.

It is my experience that the role labelling and the morpho-syntactic categorization in DUG trees never get in conflict. F-structure and C-structure, to employ the terminology of Ronald Kaplan's and Joan Bresnan's Lexical Functional Grammar, can be designed completely in parallel (as opposed to the complicated mapping between these structures in LFG). The nodes in the DUG trees display function-form units, an idea which I originally took over from Ray Pike's tagnemics.

The parallelism of function and form can easily be explained from the point of view of a word-based syntax. It is the lexical meaning of the governing word which implies complements with specific roles. The meaning of *attend* necessarily involves an attender and some event being attended; the meaning of *amaze* requires the reference to some amazing fact and to someone being amazed. As a consequence, syntactic units occur in the context of these words which are able to denote the required complements. The task of morphology is to identify these units and keep them apart from each other.

2.4 Unification

The morpho-syntactic form of syntactic units is represented in DUG by complex categories consisting of feature-types with disjunctive feature-values. The agreement of feature values is computed by comparing the templates, the potential fillers and the governing term. Intersection and union of values are the basic outcomes of the comparison. The two types of defaults of an unspecified attribute are: any value or no value applies.

Is it justified to call this computation of agreement unification, a term which originates from theorem proving? What is the essence of the unification mechanism? Technically, it means bidirectional pattern matching. Conceptually, it means that a grammar is conceived as a set of equations which are to be resolved, rather than a set of rules which are te be consecutively applied. The instantiation of the variables in the grammatical equations is provided by the lexicon. These instantiations are propagated through the equations. In my opinion, this viewpoint is an important change in the concept of grammar writing so that the label "unification grammar" is more than just a fashionable accessory.

The difference between DUG and other unification grammars is the dependency tree structure. As opposed to graph unification, the syntagmatic units are not blurred. This is a prerequisite for handling positional features. The main characteristics of a DUG model of syntax is the grouping of the syntactic information into the three dimensions of function, lexical meaning and form. Features of each dimension apply to any syntactic unit at the same time. The following diagram illustrates the feature relationships which constitute the equations to be resolved by unification in a DUG:

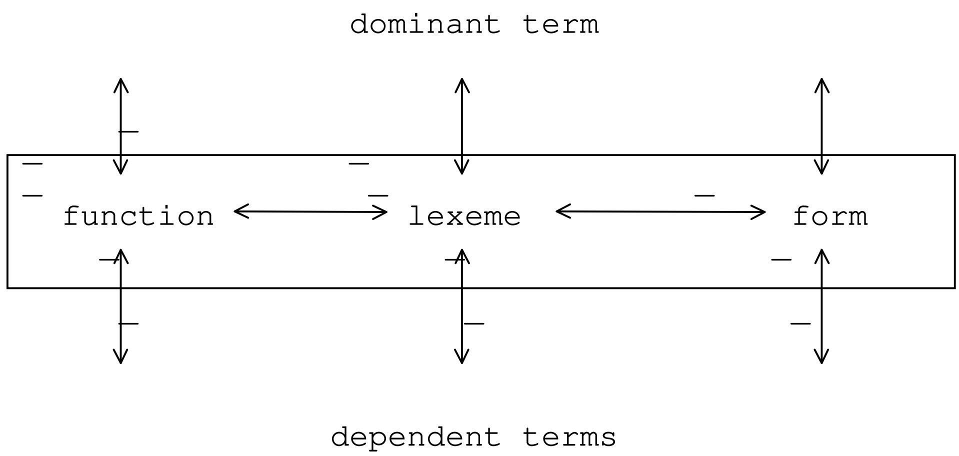

Fig. 10: Unification of functions, lexemes and forms

References

Hellwig, Peter, 1980. PLAIN - A Program System for Dependency Analysis and for Simulating Natural Language Inference. In: Leonard Bolc, ed., *Representation and Processing of Natural Language*, 271-376. Munich, Vienna, London: Hanser & Macmillan.

Hellwig, Peter, 1986. Dependency Unification Grammar (DUG). In: *Proceedings of the 11th International Conference on Computational Linguistics (COLING 86)*, 195-198. Bonn: Universität Bonn.

Hellwig, Peter, 1988. Chart Parsing According to the Slot and Filler Principle. In: *Proceedings of the 12th International Conference on Computational Linguistics (COLING 88)*, 242-244. Budapest: John von Neumann Society for Computing Sciences.