DEPENDENCY UNIFICATION GRAMMAR

Peter Hellwig
University of Heidelberg, D-6900 Heidelberg, West Germany

Abstract

This paper describes the analysis component of the language processing system PLAIN from the viewpoint of unification grammars. The principles of Dependency Unification Grammar (DUG) are discussed. The computer language DRL (Dependency Representation Language) is introduced in which DUGs can be formulated. A unification-based parsing procedure is part of the formalism. PLAIN is implemented at the universities of Heidelberg, Bonn, Flensburg, Kiel, Zurich and Cambridge U.K.

1. Introduction

The recent development of grammar theory exhibits convergencies among various approaches, such as Government-Binding Theory, Generalized Phrase Structure Grammar, Definite Clause Grammar, Lexical Functional Grammar, Functional Unification Grammar, and others. To varying degrees these theories share the following principles:

(i) They take into account dependency relations, using notions such as "head" or "governor". Phenomena such as long distance dependencies are viewed as touchstones for the formalisms.

(ii) They pay attention to functional aspects. The representation of syntactic roles is seen to be a task.

(iii) They agree that syntax must be lexically restricted and thus place a large portion of the grammatical information in the lexicon.

(iv) They base their algorithms on the principle of unification, i.e. complex categories are brought into agreement in the syntactic context.

These common features make it possible to compare the solutions of the different formalisms as well as their problems.

The main difficulty for the computer application of unification grammars lies in their complexity. LFG, for example, bases its syntactic c-structures on the phrase structure principle, while the functional f-structures represent dependency relationships between functors and arguments. This causes problems for the parser which needs information on both structures while it is creating them. The development of GPSG seems to be marked by the effort to introduce more and more types of rules so as to adequately constrain the formalism. As a result, the control of analysis is distributed over many resources and is, therefore, increasingly difficult. Since a large number of constraints are of a lexical nature, the lexicon becomes more and more unwieldy in all of the formalisms.

Common advantages and common problems of unification grammars suggest examining strategies from still other frameworks. This is to be done here with respect to dependency grammar. DUG rearranges the available means of description. As a result, the benefits of the common principles are fully felt whereas the difficulties mentioned are largely avoided.

2. Dependency Representation Language (DRL)

Grammar formalisms and computer languages are usually developed independently. DRL is both at the same time. In the same spirit as PROLOG is tailor-made for the purposes of logic, DRL has been particularly adapted to represent linguistic structures. Whereas the interpreter for PROLOG includes a theorem prover, the interpreter for DRL is linked with a parser. (DRL also serves for the purpose of knowledge representation within the deduction component of PLAIN. This aspect will not be discussed here.)

DRL consists of bracketed expressions which are lists in the sense of list processing. Conceptually, they represent tree diagrams with nodes and directed arcs. It is the characteristic feature of DRL that each node refers to a lexically defined atomic unit of an utterance and that the arcs represent direct relationships between these atomic units. According to the hierarchical structure of tree diagrams, one element in each relationship is dominant, the other one is dependent. Dependency grammar assumes that this asymmetry reflects the actual situation in natural language.

Asymmetries between constituents are commonly conceded in modern grammar theory. It seems to be certain that only via the head-complement distinction can adequate constraints for the construction of natural language expressions be defined. Unfortunately, phrase structure, which prevails in most grammar formalisms, is at odds with the direct asymmetric relations between immediate constituents. A logical consequence would be to chose dependency as the primary principle of representing syntactic structure (see the arguments in Hudson 1984). Nevertheless, this proposal still encounters quite a bit of scepticism. The implementation of an efficient parser (see Hellwig 1980) has proven the practicability of the dependency approach. However, the formalism for dependency grammars has had to be substantially augmented.

3. Factorization of Grammatical Information

When designing a computer language that is to serve as a grammatical formalism, it is crucial to provide for a factorization of information that is at the same time convenient and adequate. I have stressed that DRL terms are in a one-to-one relationship with the basic elements of a natural language. Since the features of these elements are numerous and varied, every DRL term must be multi-labeled. As is common in unification grammars, each feature is coded as an attribute-value pair. The attribute states the feature type, the values represent the concrete features. The division into attributes and values allows for very general descriptions, since relationships can now be formulated on the level of the attributes, no matter which values apply in the individual cases. A complex category consist of any number of attributes or attribute-value assignments.

Faced with the unlimited expressiveness of complex categories, the key issue now is to carefully select and group the attributes in such a way that the linguistic phenomena are represented as adequately and transparently as possible. DUG assumes that a distinction must be made among three dimensions in which each element of an utterance participates: lexical meaning, syntagmatic function and outward form. Correspondingly, three types of attributes are grouped together in each DRL-term: a lexeme, a syntagmatic role and a complex morpho-syntactic category. To give an example:

(1) The cat likes fish.

This sentence is represented in DRL as follows, disregarding positional attributes for the moment:

```
(2)  (ILLOCUTION: assertion: clse typ<1>
         (PREDICATE: like: verb fin<1> num<1> per<1>
             (SUBJECT: cat: noun num<1> per<3>
                 (DETERMINER: the: dete))
             (OBJECT: fish: noun)));
```

We cannot avoid going into a few notational details. Each term, printed on a separate line, corresponds to a word in (1). The first term is correlated to the period, which is also treated as a word. The parentheses depict the dependency structure. The first attribute in each term is the role, the second the lexeme. Both are identified by position, i.e. their values are simply written at the first and second position in the term. Roles and lexemes constitute the semantic representation. They are more or less equivalent to f-structures in LFG. The third part of each term contains a description of the surface properties of the corresponding segments in the utterance. It consists of a main category, generally a word class such as verb, noun, determiner, followed by a sequence of attribute-value subcategories which represent grammatical features such as finiteness, number, person. The format of subcategories is standardized in order to facilitate processing. Attributes are symbolized by three character-long key words, values are coded as numbers in angled brackets.

The salient point of this formalism is that the functional, the lexematic and the morpho-syntactic properties coincide in every term, as they do in the elements of natural language. To put it in the terminology of LFG: f-structure and c-structure are totally synchronized. Since this cannot be achieved in a phrase structure representation, it is often assumed that there is a fundamental divergence between form and function in natural language. Admittedly, one prerequisite for a uniform function-form correspondence still has to be mentioned. Since non-terminal constituents are not basic, they are usually not represented by terms in DRL. However, there must be something to denote the suprasegmental meaning that a clause conveys in addition to the semantics of its constituents. As a necessary extension of dependency grammar, the yield of a clause is - so to speak - lexicalized in DUG and represented by a term that dominates the corresponding list. Compare the first term in (2). Punctuation in written language can be interpreted as a similar lexicalization of clausal semantics.

4. Positional Features

An important augmentation of dependency grammar is the decision to treat positional phenomena in DUG as morpho-syntactic features and, as a consequence, to represent them by subcategories in the same way as number, person and gender. The mechanism of unification can be applied to word order attributes just as advantageously as to other categories. The only difference is that the values appertaining to the elements of an utterance are not taken from the lexicon, but are drawn from the situation in the input string. One has to visualize this as follows.

Each term in a dependency representation corresponds to a segment of the input string. Each subtree also corresponds to a segment which is composed of the segments corresponding to the terms which form the tree. Breaking down a dependency tree into subtrees thus imposes an implicit constituent structure on the input string. Incidentally, the constituent corresponding to a dependency tree does not need to be continuous. The positions of the constituents relative to each other can be determined and included as the values of positional attributes in the terms of the dependency trees. It is stipulated that a positional attribute refers to the implicit constituent corresponding to the subtree in whose dominating term the feature is specified. The attribute expresses a sequential relationship between this constituent and the segment which corresponds to the superordinated term.

Any sequential order of constituents which can be defined can be included in the set of attributes. Suppose, for example, that D is a string corresponding to a subtree and H is the string that corresponds to the term superordinated to that subtree. Let us define the attribute "sequence" (seq) as having the values 1: C precedes H, and 2: C follows H. Let us establish "adjacency" (adj) with the values 1: C immediately precedes H, and 2: C immediately follows H. Finally, let us introduce "delimitation" (lim) with the values 1: C is the leftmost of all of the strings corresponding to dependents of H, and 2: C is the rightmost of of all of the dependents of H. For the sake of comparison, let us consider the following example which Pereira 1981 uses in order to illustrate Extraposition Grammar:

(3) The mouse that the cat that likes fish chased squeaks.

The following DRL-tree depicts the dependencies and the word order of this sentence by means of the attributes just defined:

```
(4) (ILLOCUTION: assertion: adj<1>
        (PREDICATE: squeak: adj<1>
            (SUBJECT: mouse: adj<1>
                (DETERMINER: the: seq<1>)
                (ATTRIBUTE: chase: adj<2>
                    (OBJECT: that: lim<1>)
                    (SUBJECT: cat: adj<1>
                        (DETERMINER: the: adj<1>)
                        (ATTRIBUTE: like: adj<2>
                            (SUBJECT: that: lim<1>)
                            (OBJECT: fish: adj<2>)))))));
```

The projection of subtrees and their attributes yields the following constituent analysis of the input string:

```
(5) the mouse that the cat that | ←———————— squeaks
    likes fish chased           |   adj<1>

    the | ←——————— mouse ———————→ | that the cat
        |   seq<1>         adj<2> | that likes
        |                         | fish chased

    that                    | ←——————— chased
                            |   lim<1>
    the cat that likes fish | ←———————
                            |   adj<1>

    the | ←——————— cat ———————→ | that likes fish
        |   adj<1>      adj<2>  |

    that | ←——————— likes ———————→ | fish
         |   lim<1>          adj<2> |
```

There is exactly one sequence of words that is in agreement with all of the attribute-values in the tree. It is likely that appropriate attributes can also be defined for more difficult cases of extraposition. Since the dislocated elements continue to be subordinated to their heads in their original role, no "gaps", "holes" or "traces" are part of the DRL-formalism. The possibility to do without such entities is attractive. It arises from the fact that the ratio of constituency and dependency is reversed in DUG. It seems to be easier to augment dependency trees by constituency information than to process dependency features within phrase markers.

5. Morpho-syntactic Description

Within DRL terms, the following means exist for generalization. There are variables for roles, lexemes and morpho-syntactic main categories. Subcategories allow a disjunction of values as their specification. The ANY-value is assumed whenever a subcategory attribute is left out completely. These means are applied in the so-called base lexicon. The base lexicon creates the link between the segments of the input language and the terms of DRL. A few results of this assignment are to be given just to illustrate the format:

```
(6)   CAT    ->    (*: cat: noun num<1> per<3>);
      CATS   ->    (*: cat: noun num<2> per<3>);
      LIKE   ->    (*: like: verb per<1,2>);
      LIKES  ->    (*: like: verb num<1> per<3>);
      LIKE   ->    (*: like: verb num<2> per<3>);
      FISH   ->    (*: fish: noun per<3>);
```

The roles of all lexical items are left open. Their values are a matter of the syntactic frames in which the items occurs in an utterance. The same lexeme applies to all inflectional forms of a word. The values of person and number of CAT and CATS are indicated because they are specific. FISH, on the other hand, can be both singular and plural. Hence the number-attribute is omitted altogether. The features first and second person of LIKE are combined by disjunction. The choice between both values as well as between the ANY-values of number is left to the context. In case of the third person items it cannot be avoided to be more specific.

6. Slots

The notion of dependency is closely related to the idea of intrinsic combination capabilities of the lexical elements. This capability is traditionally referred to as valency, although this view has often been restricted to verbs. DUG generalizes this lexicalistic approach with respect to all syntagmatic relationships. Syntax is completely integrated in the lexicon. The natural way to state valencies is by assigning slots to possibly dominating terms. A slot is a template of the list that would be an appropriate complement. As a rule, only the head of this list has to be described, because head-feature-convention (as known from GPSG) is a general principle in dependency representation. The following is a description of the valency of LIKES:

```
(7)   (*: like: verb fin<1> num<1> per<3>
         (SUBJECT: _ : noun num<1> per<3> adj<1>)
         (OBJECT: _ : noun seq<2>));
```

Slots are the places where roles are introduced into the formalism. As a matter of fact, it is the task of roles to differentiate complements. The lexematic character of the complements is usually unrestricted and, therefore, represented by a variable. Morpho-syntactics categories express the formal requirements, including positional attributes, that the filler must meet.

A direct assignment of slots to a specific lexical item is good policy only in the case of idiosyncratic complements. Complements such as subject and object that are shared by many other verbs should be described in a more general way. The solution is to draw up completion patterns once and to refer to those patterns from the various individual lexemes. A separate pattern should be set up for each syntagmatic relationship. For example:

```
(8)   (*: +subject: verb fin<1>
         (SUBJECT: _ : noun num<C> per<C> adj<1>));

(9)   (*: +object
         (OBJECT: _ : noun seq<2>));
```

The following entries in the valency lexicon illustrate references to these patterns:

```
(10)  (: -> (*: squeak)  (: +subject));

(11)  (: -> (*: like)    (& (: +subject)
                            (: +object)));
```

In the case of LIKES, the effect of (11) is identical to (7).

Certain provisions allow for a maximal generality of patterns. The symbol "C" as subcategory value in (8) indicates that the respective values of a potential filler and the head of the list must match whatever these values may be in the concrete case. Hence, pattern (8) covers subjects with any number and person features and, at the same time, controls their agreement with the dominating predicate. Morphological features in the head term restrict the applicability of the pattern. In the case of (8) the dominating verb must be finite (fin<1>), because it cannot have a subject as complement in the infinitive. The object pattern, on the contrary, is applicable without restrictions.

An analogy to feature disjunction on the paradigmatic level is slot disjunction on the syntagmatic level. It is the means to formalize syntactic alternatives. The following improved patterns for subjects and objects include slots for relative pronouns in their appropriate leftmost position:

```
(12) (*: +subject: verb fin<1> per<3>
        (, (SUBJECT: _ : pron rel<1,C> lim<1>)
           (SUBJECT: _ : noun num<C> per<C> adj<1>)));

(13) (*: +object
        (, (OBJECT: _ : pron rel<1,C> lim<1>)
           (OBJECT: _ : noun seq<2>)));
```

(12) provides for "that likes fish" and (13) for"that the cat chased" in Pereira's example. The feature "rel<1>", which is intrinsic to the relative pronoun, is to be passed on to the dominating verb as is indicated by "C". This is the prerequisite to identifying the verb as the head of a relative clause. The pattern for the relative clause could look like this:

```
(14) (*: +relative clause: noun
        (ATTRIBUTE: _ : verb rel<1> fin<1> adj<2>));
```

The following patterns and references complete the small grammar that is needed for Pereira's sentence:

```
(15  (*: +determiner: noun
       (DETERMINER: _ : dete seq<1>));

(16) (: -> (*: mouse)   (&  (: +determiner)
                            (: +relative clause)));
(17) (: -> (*: cat)     (&  (: +determiner)
                            (: +relative clause)));

(18) (ILLOCUTION: assertion: clse typ<1>
       (PREDICATE: _ : verb fin<1> adj<1>));
```

Completion patterns capture the same syntactic regularities as rules in other formalisms. The peculiarity of DUG is that it breaks down the complex syntax of a language into many atomic syntactic relationships. This has several advantages. Valency descriptions are relatively easy to draw up. They are to a great extent independent of each other so that changes and additions normally have no side effects. Although the grammar is wholly integrated in the lexicon, the structure of lexical entries is rather simple. Any new combination of complements which may be encountered is simply a matter of lexical reference, while in rule-based grammars a new rule has to be created whose application subsequently has to be controlled.

7. Parsing by Unification

In logic, unification is defined as a coherent replacement of symbols within two formulas so that both formulas become identical. The same principle can be applied advantageously in grammar. The basis of the mechanism is the notion of subsumption. There are two occurrences of subsumption in DRL. Firstly, attribute symbols subsume all of the appertaining values. For example, a role variable covers any role, a morpho-syntactic subcategory covers any element of the defined set of features. Secondly, structure descriptions subsume structures. DRL comprises variables which refer to various substructures of trees. In the present context we consider only direct subordination of slots.

It must be the strategy of the grammar writer to keep any single description as abstract as possible so that it covers a maximum number of cases. In the course of the analysis, the unification of expressions leads to the replacement of the more general by the more specific. As opposed to simple pattern matching techniques, replacements of the symbols of two expressions occur in both directions. Continued unification in the syntagmatic framework leads to an incremental precision of the attributes of all of the constituents.

A prerequisite for a unification-based parser is the control of the expressions which are to be unified. The control structure depends on the grammar theory which is at the basis. The PLAIN parser runs through three phases: (i) the consultation of the base lexicon yielding a lexeme and a morpho-syntactic characterization for each basic segment in the utterance, (ii) the consultation of the valency lexicon yielding a description of the combination capabilities of the basic terms, (iii) a reconstruction of the syntactic relationships in the utterance by a bottom-up slot-filling mechanism. Throughout the whole process previous expressions are unified with subsequent ones.

Let us first consider the lexicon phases. The word forms in the utterance are taken as the starting points. According to the base lexicon, they are replaced by terms which show the identity and the divergence of their attributes. With respect to identity, this is a step similar to unification. Compare, for example, the terms associated with the word forms of "to like" in (6), which share the role, the lexeme and the word class properties. With respect to divergence, the base lexicon contains just as many features as can be attributed to a word form out of the syntactic context. The valency lexicon, on the other hand, abstracts just from those features which are not distinctive for a specific syntactic relationship.

The parser combines the information from both lexica by means of unification. At first, the terms derived from the base lexicon are unified with the left-hand side of the valency references. The resulting specification is transferred to all terms on the right-hand side of the reference. Each of these terms, in turn, is unified with the heads of the completion patterns. The specifications produced in the course of these operations are brought into agreement with the original terms and, eventually, the appropriate slots are subordinated to these terms.

Once the initial lists are produced comprising the combined information from both lexica, the detection of the syntactic structure of the utterance is a fairly simple process. Each of the lists, starting with the leftmost one, tries to find a slot in another list. If a searching list can be unified with the attributes in a slot, a new list is formed which comprises both lists as well as the result of their mutual specifications. The new list is stored at the end of the line and, when it is its turn, looks for a slot itself. This process continues until no more lists are produced and no slots are untried. Those lists that comprise exactly one term for each input segment are the final parsing results.

I would like to stress a few properties that this parsing algorithm owes to DUG. Similar to unification in theorem proving, the process relies completely on the unification of potential representations of parts of the utterance. No reference to external resources, such as rules, taint the mechanism. The control is thus extremely data- directed. On the other hand, the unification of DRL lists is an instrument with an immense combinatorial power. Within any term the agreement of function, lexical selection and morpho-syntactic features is forced. In addition to this horizontal linkage, the attributes of the dominating term as well as the attributes of the dependent terms are also subject to unification. The attributes of dependent terms are delineated by the valency description. According to congruence conditions heads and dependents continue to mutually specify each other. Feature unification and slot disjunction also restricts the co-occurrence of dependents. In addition, positional features are continuously made to tally with the corresponding sequence of segments in the input string. This network of relationships prevents the parser from producing inappropriate lists. At the same time it results in incremental specification, which facilitates the work of the lexicon writer. What may be theoretically the most interesting is the fact that functional, lexical, morphological and positional features can be processed smoothly in parallel.

References

Hellwig, P.: "PLAIN - A Program System for Dependency Analysis and for Simulating Natural Language Inference." In L. Bolc (ed.): Representation and Processing of Natural Language. London: Macmillan 1980. 271-376.

Hudson, R.: Word Grammar. Oxford: Blackwell 1984.

Pereira, F.: "Extraposition Grammars." American Journal of Computational Linguistics 7 (1981). 243-256.