CHART PARSING ACCORDING TO THE SLOT AND FILLER PRINCIPLE

Peter HELLWIG

University of Heidelberg
P.O. Box 105 760
D-6900 Heidelberg, FRG

Abstract

A parser is an algorithm that assigns a structural description to a string according to a grammar. It follows from this definition that there are three general issues in parser design: the structure to be assigned, the type of grammar, the recognition algorithm. Common parsers employ phrase structure descriptions, rule-based grammars, and derivation or transition oriented recognition. The following choices result in a new parser: The structure to be assigned to the input is a dependency tree with lexical, morpho-syntactic and functional-syntactic information associated with each node and coded by complex categories which are subject to unification. The grammar is lexicalized, i.e. the syntactical relationships are stated as part of the lexical descriptions of the elements of the language. The algorithm relies on the slot and filler principle in order to draw up complex structures. It utilizes a well-formed substring table (chart) which allows for discontinuous segments.

1. Dependency Structure

The structuring principle of constituency trees is concatenation and the part-whole relationship. The structuring principle of dependency trees is the relationship between lexemes and their complements. Note: It is not correct (or at least misleading) to define dependency as a relationship between words, as it is often done. The possibility and necessity of complements depend on the lexical meaning of words, i.e. a word which denotes a relationship asks for entities which it relates, a word which denotes a modification asks for an entity which it modifies etc. While it is awkward to associate functions (deep cases, roles, grammatical relationships) with phrase structures, it is not difficult to paraphrase the functions of complements on a lexical basis. For example, the argument of the predicate "sleep" denotes the sleeper; the meaning of "persuade" includes the persuader, the persuaded person and the contents of the persuasion. In a next step, one can abstract from the concrete function of dependents and arrive at abstract functions like subject, object, adjunct etc.

Of course, the complements covering these roles can be single words as well as large phrases; for example "John", "my father", "the president of the United States" can all fill the role of the sleeper with respect to the predicate "sleep". However, phrases need not be represented by separate nodes in dependency trees (as they do in phrase markers) because their internal structure is again a question of dependency between lexemes and their complements. In a dependency tree, phrases are represented directly by their internal structure, which results in an arc between the superodinated head and the head within the complementary phrase. Nevertheless, the real principle of dependency is a relationship between words and structures, or, formally, between single nodes and trees. Taking this into account, dependency trees are much more appealing than has been recognized so far.

In order to restrict linguistic structures according to syntactic and semantic requirements, the use of complex categories is state of the art. Complex categories are sets of parameters (attributes) and values (features). Agreement between entities can be formulated in a general way in terms of parameters; the assignment of actual feature values is achieved by the mechanism of unification. If dependency is the relationship along which the catagories are unified, functional-syntactic and morpho-syntactic features can be handeled completely in parallel, as opposed to the two-phase mechanism which, for example, characterizes Lexical Functional Grammar. Each element in the dependency tree carries three labels: a role (which applies to the (sub)tree of which the element is the head), a lexeme, and a set of grammatical features.

Constituency and dependency both have to be represented somehow or other in the syntactic description. As a consequence, recent developments have led to a convergence of formalisms of both origins with respect to their contents. (A good example is the similarity between Head-Driven Phrase Structure Grammar /Pollard, Sag 1987/ and Dependency Unification Grammar /Hellwig 1986/.) If phrase structure trees are used, the difference between governor and dependent must be denoted by the categories that label the nodes, e.g. by a x-bar notation. If dependency trees are used, the concatenation relationship must be denoted by positional features which are part of the complex morpho-syntactic category.

2. Chart parsing based on a lexicalized grammar

The structure to be associated with a well-formed string can be defined in two ways: either by a set of abstract rules which describe the possible constructions of the language or by a description of the combination capabilities of the basic elements. The latter fits with the dependency approach. Given a lexical item and its morpho-syntactic properties, it is relatively easy to give a precise description of its possible complements. The main advantage of this lexicalistic approach is the fact that augmenting or changing the description of an item normally does not interfere with the rest while any change in a rule-based grammar might produce unforeseen side effects with regard to the whole.

The prerequisite for a lexicalized dependency grammar are trees that comprise slots. A slot is a description of the head of a tree that fits into another tree. Formally, a slot is a direct dependent of a head with a role associated to it, with a variable in

the lexeme position, and with a categorization that covers all of the morpho-syntactic properties of the appertaining complement. If cross categorization does not allow all of the possible properties of a complement within one category to be stated, a disjunction of slots is used to express the alternatives. The only mechanism needed for drawing up complex structures is the unification of slots and potential fillers.

The control of the parsing process is achieved by means of a well-formed substring table (chart). It is widely accepted that chart parsing is superior to backtracking or to parallel processing of every path. A common version of a chart can be vizualized as a network of vertices representing points in the input, linked by edges representing segments. The edges are labelled with the categories that the parser has assigned to the constituents concerned. Alternatively, each edge is associated with a complete structural description, including the information which is carried by the covered edges. In this case, a chart is simply a collection of trees (implemented as lists) projected on the various segments in the input. The innovation with regard to chart parsing that is proposed in this paper is a labelling of edges by trees that comprise slots.

At the beginning, an edge for each word is entered into the chart. Each edge is labelled with a tree. The head of this tree contains the lexeme that is associated with the word according to the lexicon; it carries a morpho-syntactic category according to the morphological properties of the word in question; it normally contains a variable as a role marker, since the syntagmatic function of the corresponding segment is still unknown. A slot is subordinated to the head for each element that is to be dependent in the resulting structure, if any. Slots are added to a lexical item according to completion patterns refered to in the lexicon. (We can not go into details here.)

Subsequently, each tree in the chart looks for a slot in a tree that is associated with annother edge. If the head of the searching tree fits the description in the slot then a new edge is drawn and labelled with the compound tree that results from inserting the first tree into the second. The categories of the new tree are the result of unifying the categories of the slot tree and the filler tree. Special features state the positional requirements, e.g. whether the segment corresponding to the filler has to preceed or to follow of the segment corresponding to the element dominating the slot. This process continues until no new tree is produced. Parsing was successful if at least one edge covers the whole input. The dependency tree associated with this edge is the desired structural description.

The following example illustrates the mechanism.

(1) Flying planes can be dangerous.

The lexicon lookup leads to the initialization of the chart in figure 1.

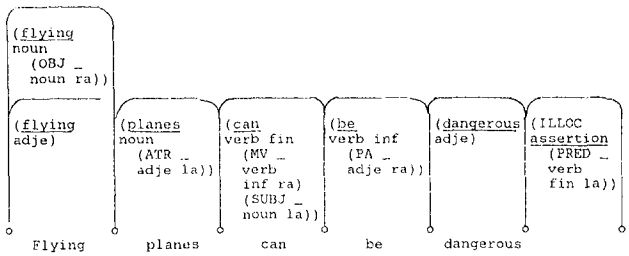

Fig. 1 The chart at the end of the lexicon phase

(The notation is simplified and we have omitted a detailed categorization. The dependency structure is represented by the bracketing. Lexemes are underlined; roles are in capital letters; slots are marked by the underscore in the lexeme position. "ra" means right adjacent, "la" means left adjacent with respect to the head.)

At the end of the slot filling process the chart looks like figure 2.

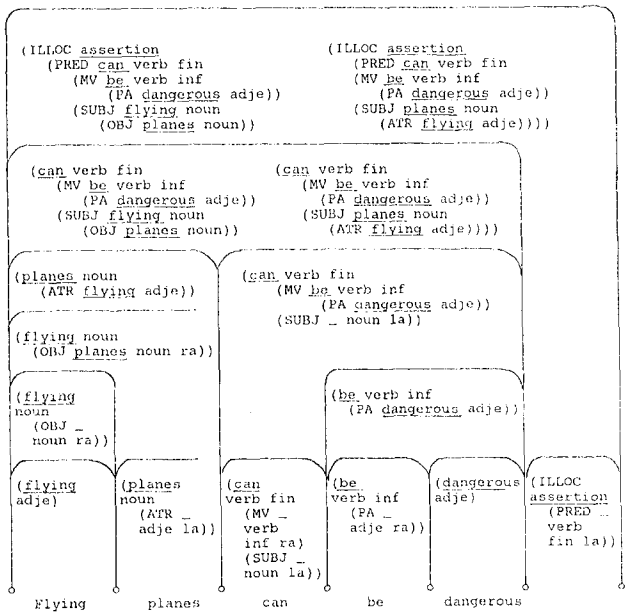

Fig. 2 The chart at the end of the parsing process

3. A chart for discontinuous segments

In figure 2, the vertices represent left and right margins of segments. Adjacency is the inherent positional relationship in this model. As a consequence, this chart does not allow for discontinuous constituents. This is a serious deficiency considering the phenomena of unbound dependencies which occur in many languages. The representation of position by categories, however, opens up the possibility to state various kinds of positional relationships between heads and dependents. In order to form discontinuous segments, the control of positions in the chart must be altered. The solution which we adopt is to represent the extensions of segments by bit strings consisting of one bit

for each word that is part of the segment and a zero bit for any word that is not within the segement (cf. fig. 3, trailing zeros are omitted). Discontinuous segements are depicted by the alternation of ones and zeros in the bit string. The intersection of the appertaining bit strings is formed before a filler is compared with a slot. This intersection must be empty, because otherwise both segments contain overlapping portions and one can not be a dependent of the other. After a filler is inserted into a slot, the union of both bit strings is formed and associated with the new tree. This amounts to island extension of the parsed substrings. Other operations on the bit strings allow for calculating the specific positional relationships stated within the categories of slots. The parser runs through alternate cycles: It tries to build continuous segments first. Then it uses these segments to form discontinuous segments. From the results it tries to build continuous segments again and so on. The input is accepted if a bitstring can be obtained that contains as many bits as there are words in the input.

The example in figure 3 roughly illustrates the device. (Integers among the categories represent the sequence of dependents belonging to the same head. Lines separate the cycles.)

(2) What does Gudrun feed to her cat ?

```
     Position: Tree:

(1) 1          (what pron)
(2) 01         (do verb fin
                  (SUBJ _ noun ra 1)
                  (MV _ verb inf 2))
(3) 001        (Gudrun noun)
(4) 0001       (feed verb inf
                  (DOBJ _ 1)
                  (IDOBJ _ to ra 2))
(5) 00001      (to
                  (_ noun ra))
(6) 000001     (her poss)
(7) 0000001    (cat noun
                  (DET _ poss 1a))
(8) 00000001   (ILLOC question
                  (PRED _ do verb fin 1a))
(9) 011        (do verb fin
                  (SUBJ Gudrun noun ra 1)
                  (MV _ verb inf 2))
(10) 0000011   (cat
                  (DET her poss 1a))
(11) 0000111   (to
                  (cat noun ra
                     (DET her poss 1a)))
(12) 0001111   (feed verb inf
                  (DOBJ _ 1)
                  (IDOBJ to ra 2
                     (cat noun
                        (DET her poss 1a))))
-------------------------------------------------------------
(13) 1001111   (feed verb inf
                  (DOBJ what 1)
                  (IDOBJ to ra 2
                     (cat noun
                        (DET her poss 1a))))
-------------------------------------------------------------
(14) 1111111   (do verb fin
                  (SUBJ Gudrun noun ra 1)
                  (MV feed verb inf
                      (DOBJ what 1)
                      (IDOBJ to ra 2
                          (cat noun
                              (DET her poss 1a)))))
(15) 11111111  (ILLOC question
                  (PRED do verb fin
                      (SUBJ Gudrun noun ra 1)
                      (MV feed verb inf
                          (DOBJ what 1)
                          (IDOBJ to ra 2
                              (cat noun
                              (DET her poss 1a)))))
```

Fig. 3 States of a chart including discontinuous segments

The parser is fully implemented within the system PLAIN (Programs for Language Analysis and Inference), developed at the University of Heidelberg. So far, it is applied to German (Heidelberg, Kiel), French (Paris), and English (Surrey/UK, Pisa, Hawaii).

References

Peter Hellwig: "PLAIN - A Program System for Dependency Analysis and for Simulating Natural Language Inference". In: Leonard Bolc (ed.): Representation and Processing of Natural Language. Wien: Hanser; London, Basingstoke: Macmillan 1980, pp. 271-376.

Peter Hellwig: "Dependency Unification Grammar". In: 11th International Conference on Computational Linguistics. Proceedings. Bonn, August, 25th to 29th, 1986, University of Bonn, pp. 195-198.

Carl Pollard, Ivan A. Sag: Information-Based Syntax and Semantics. Vol. 1. Fundamentals. CSLI Lecture Notes, No. 12., Stanford University 1987.