GRAPH GRAMMAR APPROACH TO NATURAL LANGUAGE PARSING AND UNDERSTANDING

Eero Hyvonen

Digital Systems Laboratory
Helsinki University of Technology
02150 Espoo 15, Einland

ABSTRACT

String grammars have been found in many ways inadequate for parsing inflectional languages with "free" word order. To overcome these problems we have replaced linear string grammars and tree transformations by their multidimensional generalization, graph grammars. In our approach parsing is seen as a transformation between two graph languages, namely the sets of morphological and semantic representations of natural language sentences. An experimental Finnish question-answering system SUVI based on graph grammars has been implemented. In SUVI the role of individual words is active. Each word is associated to a syntactico-senantic constituent type that is represented by a transition network -like graph whose transitions correspond to transformations in the derivation graph. Parsing is performed by interpreting the constituent type graphs corresponding to the words of the current sentence.

1 WHY GRAPH GRAMMARS ?

In parsing highly inflectional languages with loose syntax by string grammars several problems arise with the formalism:

1) Morphological structures cannot be adequately represented by trees and string grammars.
2) Many relations between words and constituents are difficult to express by trees and string rewrite rules. For example, case, person, and number agreements are widely used.
3) Relatively free word and constituent ordering found in inflectional languages leads to large and complex grammars with string grammars that are based on linear ordering of symbols.
4) Discontinuous constituents occur often in inflectional languages.
5) The levels of morphology, syntax and semantics are quite intermingled in inflectional languages. For example, in Finnish the correlation between morphological and semantic cases is essential.

To solve these problems a stronger, multidimensional formalism that can cope with different levels of language seems necessary. In this paper we suggest graph grammars /1, 2, 3/ for the purpose. Besides their high generative power graph grammars offer an illustrative formal tool for modelling language understanding processes.

2 A GRAPH GRAMMAR FORMALISM FOR NATURAL LANGUAGE PROCESSING

Algebraic graph grammars (AGG) /4/ can be considered a multidimensional generalization of linear string grammars. In AGG's the catenation of strings is replaced by the more versatile notion of "gluing" graphs together. String productions are generalized into graph productions which explicitly express structural transformations in the derivation graph. Applying a graph production means that a subgraph corresponding to the left hand side of the production is replaced by its right hand side in the derivation graph. However, additional "application conditions" must hold to quarantee that arcs with no source or goal node will not arise and that the result of the derivation is well-defined. Analogously to the string case a graph language is defined as the set of all terminally colored graphs derivable from an initial start graph by some sequence of productions of the grammar.

The formalism of this paper /1/ contrasts the basic notions of AGG's in three major ways:

- Grammatical derivation sequences are given explicitly by transition network -like "control graphs" whose transitions correspond to transformations in the derivation graph (cf. the notion of "programmed graph grammars" /6/).
- Application conditions in our grammars are trivially true due to some restrictions and modifications to AGG's. The resulting formalism is argued to be intuitive, semantically plausible, and computationally efficient.
- Nodes and arcs in the graphs are associated to a set of properties that can be modified as a side effect of direct derivations (cf. the notion of "attributed graph grammars" /7/).

In our experimental question-answering system SUVI nodes have three properties: EXT (extension) is the set of all objects in the real world knowledge base the node currently refers to; QUANT is the quantifier of the extension; CAT is the syntactico-Bemantic type or category of the

node. Arcs have no properties. Essentially, the derivation graph is a set of relations between quantified extensional sets of grammatical and real world objects belonging to different syntactico-semantic categories.

3 A GRAPH GRAMMAR BASED SEMANTIC PARSER

Figure 1 depicts the function and structure of our SUVI-systern.

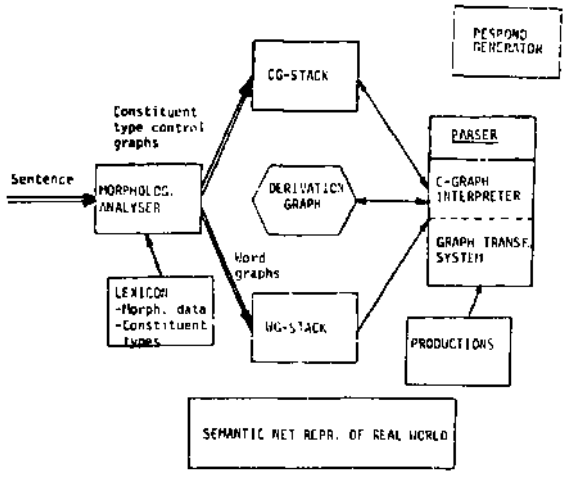

Figure 1. The overall structure of SUVI and the graph grammer parser.

First words and idioms are delinearized by the morphological analyzer into small frame-like word graphs representing morphological word forms. Initial node properties (EXT, QUANT, and CAT) are associated to the nodes. For example, the singular partitive noun "ohjelmaa" (a program) would refer existentially to a program in the set of all programs while the nominative plural form "ohjelmat" (the programs) refers to all programs.

The upper graph of figure 2 illustrates part of a derivation graph that contains the morphological structure of three successive Finnish words "kaikki suuret ohjelmasi" (all big programs-of-yours). The last word "ohjelmasi" is morphologically ambiguous but can be represented by a single word graph by setting non-unary extensions to nodes expressing the case and number. These extensions will later be automatically focused in clause context by direct derivations demanding congruences between words and constituents. Entries and word forms having different meaning or syntactico-semantic properties are considered lexically ambiguous. They are represented by a set of alternative word graphs.

The goal of the parser is to generate from each morphological representation of the sentence corresponding hierarchical semantic case structure. The parser has three major dynamic data structures (fig. 1): the derivation graph, the stack of word graphs (WG-STACK), and the stack of transition network -like control graphs (c-graphs) with priorities (CG-STACK). After morphological analysis CG-STACK consists of the control graphs corresponding to the constituent types of the individual, words in the sentence and a set of sentence-independent transformational graphs. Arcs in c-graphs usually correspond to direct derivations in the derivation graph but can perform other operations, too, like pushing new c-graphs into the CG-STACK to be interpreted later. By this way, for example, nouns analyse their post-attributes and verbs their deep case constituents.

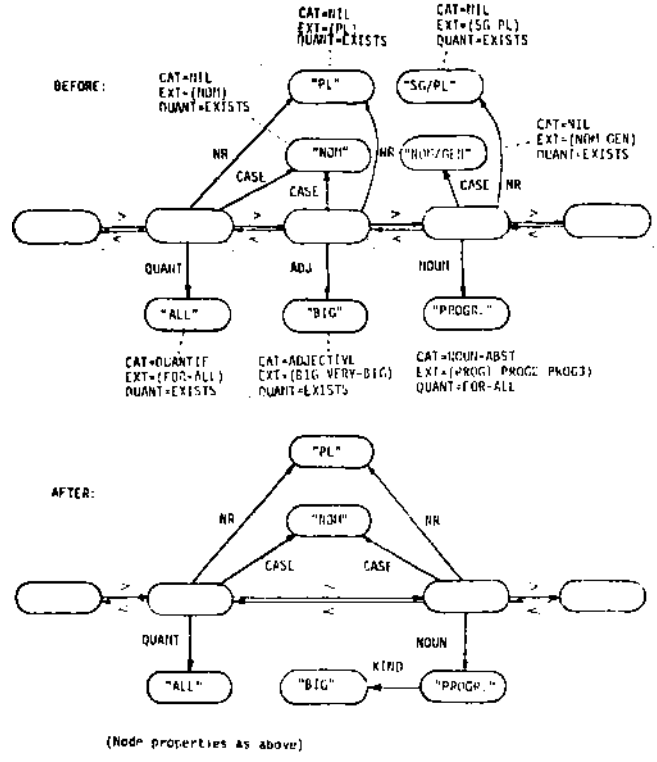

Figure 2. Part of a derivation graph before and after direct derivation ADJ-ATTR.

Parsing begins by interpreting the first c-graph in the CG-STACK. The constituent type c-graphs are defined in such a way that all word graphs from the WG-STACK will eventually be glued into the derivation graph and corresponding c-graphs interpreted. By the transformational c-graphs passive clauses and yes/no-questions are transformed into simple active declaratives. In the end CG-STACK and WG-STACK are empty and the derivation graph constitutes a labelled tree of semantic case propositions.

C-graphs are illustrative descriptions of syntactico-semantic constituent types. For example, figure 3 tells that in an NP-HUMAN constituent genetive, pronoun, and adjective attributes with a quantifier may precede the main word in the combinations expressed by the graph. Attributes can be already parsed lower level constituents as well as single words, because these both are represented in a compatible way.

In nodes with a dot inside backtracking is allowed if some of the out-going arcs marked by a question mark have first been chosen. By this way locally ambiguous syntactic structures are analysed.

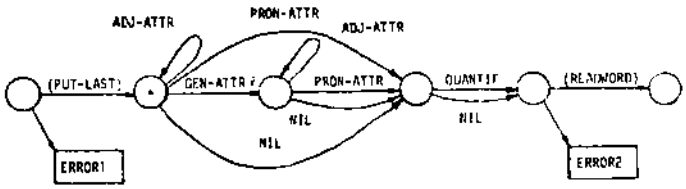

Figure 3. The control graph of constituent type NP-HUMAN.

Production ADJ-ATTR in figure 4 illustrates the structure and meaning of the adjective attribute construction in Finnish (cf. linear production AdjP+NP->NP). Uncolored nodes in the left hand side match any node with no side effect. The interpretation of the production is that if there is an adjective followed "by a noun in the same case and number in the sentence the noun and the adjective (their extensions) are assumed to be in semantic Kind-relation with each other in the knowledge base. figure 2 illustrates the application of ADJ-ATTR production in a derivation graph. The case and number of the noun "PROGR." are well-defined after the derivation due to the congruence demand of the adjective attribute construction.

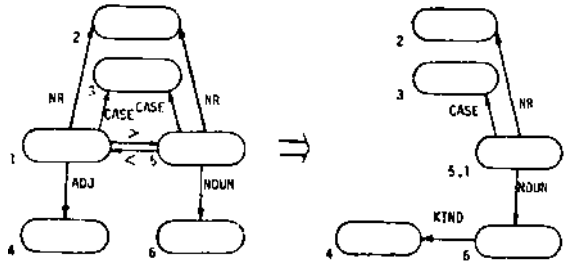

Figure 4. Production ADJ-ATTR for the adjective attribute construction.

At the highest level of constituent generation in a clause — i.e. when interpreting the c-graph with the lowest priority in the CG-STACK — semantic deep case relations are generated between the predicate verb and its deep case constituents. Figure 5 illustrates a production for generating the AG-case. Node <NOUN-HUM.> matches nodes with CAT=NOUN-HUMAN and NOM,PART matches nodes expressing nominative or partitive case. Arc <,> matches both <- and >-arcs.

Figure 6 depicts the resulting semantic representation for the question

"Kuka laboratorion luennoitsijoista on luennoinut jonkun serainaarimaisen kurssin tietojerikasittelyteoriasta syksylla 1981 ?"
(Which lecturer of the laboratory has lectured some seminar-type course on computer science in the autumn 1981 ?).

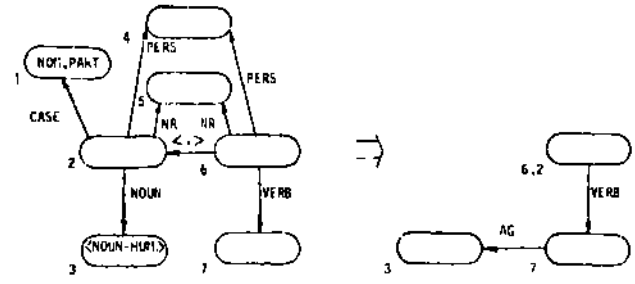

Figure 5. A production for generating the AG-case in the derivation graph.

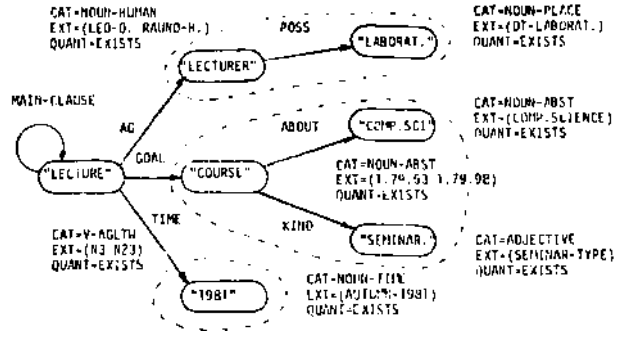

Figure 6. Semantic representation of a sentence with three major constitupnts.

Dotted lines encircle the semantic case representations of the three major constituents of the clause. The case structures are generated by corresponding constituent type c-graphs.

After parsing RESP0ND-GENERATOR computes the extensions of the constituents in the clause context. This is obtained by recursively demanding that the semantic cases between the constituent nodes in the derivation graph hold for their extensions in the knowledge base. The final extension of the MAIN-CLAUSE-node corresponding to the predicate of the sentence is a set of propositional deep case structures in the knowledge base that satisfy the semantic conditions expressed by the derivation graph. By comparing these propositions with respect to the quantified extensions of the deep case constituent nodes it is possible to generate answers to different kind of questions and requests.

As a whole, the idea of our graph grammar parser is to construct a sentence-dependent graph grammar and apply it to the morphological representations) of the current sentence. The control mechanism to find the correct derivation sequence(s) to corresponding semantic case structure(s) is essentially defined by the constituent types of individual words. Parsing is seen as a meaning preserving transformation between two graph languages — i.e. the sets of morphological and semantic representations of sentences. These graph languages correspond to two representational levels of the natural language to be parsed.

5 THE ADVANTAGES OF THE GRAPH GRAMMAR PARSER

In this paper graph grammars have been introduced into the field of natural language processing. As a conclusion, let us list what we consider important points from theoretical and practical point of view in our graph grammar approach:

1) Expressive power. Using graph grammars any kind of labelled relations can be represented and processed, not only sequential and hierarchic ones. The same formalism can be used uniformly at different levels of language.
2) Designer friendliness. Graphical drawings support design process in the same way as blue prints in many fields of engineering and offer the basis for documentation. The conversion from pictures to data structures is straight forward.
3) Uniformity. Graphs can be used to express different kinds of procedural, and declarative knowledge like c-graphs and semantic networks.
4) Extendability and modifiability. In SUVI constituent models can be designed and modified modularily by c-graph descriptions. We have asked the question What is in the word ?' a bit in the same spirit as in /8, 9/. A major point in which our parser contrasts the "word expert" parsers is that its function is specified by explicit syntactic(o-semantic) models of constituents.
5) Computational efficiency. Specialized CG-STACK's are created dynamically for individual sentences. Only relevant language knowledge is used during parsing. We argue that productions and grammars of our type can be designed to be efficient /5/.

Due to these issues the graph grammar approach seems promising in handling the problems 1-5 of parsing inflectional language as listed in section 1:

1) Complex morphological structures and many ambiguities can be expressed intuitively by word graphs.
?.) Figures 5 and 6 illustrate rewrite rules containing multiple congruences between two words or constituents.
3) The production of figure 6 shows a way of representing free constituent order; the agent noun phrase can be situated either before or after the predicate.
4) Problems of representing discontinuous constituents can be approached with productions in which congruences, category types etc. properties are demanded in stead of sequential relations. We have used this kind of productions in analysing allied verb forms.
5) Morphological, syntactic, and semantic categories and relations are used in parallel in a single production (fig. 5 and 6). However, SUVI identifies the difference between structurally and semantically (domain-dependently) incorrect sentences since domain-dependent criteria have not been used to quide parsing.

Graphical representations have traditionally been used to illustrate different kinds of linguistic and computational phenomena. Our intuition is that they should be used not only as illustrations but as a discipline and a tool in designing and implementing models on these phenomena.

6 ACKNOWLEDGEMENTS

Thanks to the personnel of Digital Systems Laboratory and Jouko Seppanen for fruitful discussions and proof reading earlier versions of this paper. Finnish Academy and the Finnish Cultural Foundation have supported our work financially.

7 REFERENCES

/1/ Hyvonen E.: A Graph Grammar Formalism and Procedural Production Description System for Natural Language Processing. Helsinki University of Technology, Digital Systems Laboratory, report B25.

/2/ Claus V., Ehrig H., and Rozenberg G. (Eds): Graph Grammars and Their Aplication to Computer Science and Biology. Lecture Notes in Computer Science 73, Springer-Verlag 1979.

/3/ Nagl, M.: Graph-grammatiken, Friedr. Verlag et Sohn, Braunschweig, 1979.

/4/ Ehrig H.: Introduction to the Algebraic Theory of Graph Grammars. Article in /2/, pp. 1-69.

/5/ Hyvbnen E.: Graph Grammar Approach to Parsing and Understanding Inflectional. Languages. A forthcoming dissertation, Helsinki University of Technology, Digital Systems Laboratory.

/6/ Bunke H.: Programmed Graph Grammars. Article in /2/, pp. 155-166.

/7/ Bunke II.: Attributed Programmed Graph Grammars and Their Application to Schematic Diagram Interpretation. IEEE Transactions of Pattern Analysis and Machine Intelligence. Vol PAMI-4, No 6, 1982.

/8/ Rieger C, Small S.: Parsing and Comprehending with Word Experts (A Theory and Its Realization). Report TR-1039, University of Maryland, Comp. Sci. Dept., 1981.

/9/ Riesbeck C: Computational Understanding: Analysis of Sentences and Context. Ph.D Thesis, Comp. Sci. Dept., Stanford University, 1974.