Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

73

Graph-Grammars and Their Application to Computer Science and Biology

International Workshop

Edited by
Volker Claus, Hartmut Ehrig and Grzegorz Rozenberg

Springer-Verlag
Berlin Heidelberg New York

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

Graph-Grammars and Their Application to Computer Science and Biology

International Workshop
Bad Honnef, October 30 – November 3, 1978

Edited by
Volker Claus, Hartmut Ehrig and Grzegorz Rozenberg

Springer-Verlag
Berlin Heidelberg New York 1979

Editors
Volker Claus
Universität Dortmund
Lehrstuhl Informatik II
Postfach 500500
D-4600 Dortmund 50

Hartmut Ehrig
Technische Universität Berlin
Fachbereich Informatik
Otto-Suhr-Allee 18/20
D-1000 Berlin 10

Grzegorz Rozenberg
Rijksuniversiteit te Leiden
Subfaculteit der Wiskunde
Wassenaarseweg 80
Postbus 9512
2300 RA Leiden

AMS Subject Classifications (1970): 68-00, 68A30
CR Subject Classifications (1974): 4.0, 5.0

ISBN 3-540-09525-X Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-09525-X Springer-Verlag New York Heidelberg Berlin

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.
2145/3140-543210

Preface

One of the older and by now well-established areas of theoretical computer science is formal language theory. Roughly speaking it deals with finite specifications of (possibly infinite) sets of strings. The sets of strings (called string languages) turn out to be useful for describing quite a variety of phenomena in a number of disciplines of science such as for example linguistics, computer science, engineering, psychology and biology. However in many applications of formal language theory applying string languages was considered to be the first step only, leading to a more general theory where sets of multidimensional objects, as well as various tranformations of them could be (finitely) described and studied. Then for example various problems in data bases, semantics of programming languages, two-dimensional programming languages, data flow analysis or incremental compilers call for finite grammatical definitions of sets of graphs, whereas various problems in picture processing and biological pattern generation require finite grammatical (or machine) descriptions of sets of maps. Extending the theory of formal (string) languages to a theory of formal multidimensional languages is a very natural step from the mathematical point of view.

Consequently there is a need, justified by both practical and theoretical considerations, to built up a theory of languages able to accomodate structures more general than strings - for example graphs and maps. Indeed various efforts were made in this direction within the last 10 years. However it is clear that the theory available (which we loosely refer to as "graph grammar theory" for historical reasons) so far does not match the theory of formal string languages. One obvious reason for this is that from the mathematical point of view the sets of structures like graphs and maps are intrinsically more difficult to deal with than the sets of strings. However another reason may be that in spite of the genuine interest in the topic there is a lack of concentrated effort in building up the desired theory. The "graph grammar community" appears to be quite scattered, not communicating well with each other and not communicating its findings very well to the outside world.

It was our perception of this lack of communication that gave us the idea to organize a meeting of researchers from very diversified areas of science who are either active workers in the theory of graph grammars or have a genuine interest in this area. This meeting took place in Bad Honnef, West Germany, in the fall of 1978 and the present

collection is a direct consequence of this meeting. Not all of the papers presented at the meeting appear in this volume: some of them were already committed somewhere else, others did not make it through the selection process. Also some papers from this volume were not presented at the above mentioned meeting, but in our opinion their inclusion gives a better view of the current state of art in graph grammar theory. Of special character are the two first papers of this volume together presenting a rather complete survey of graph grammar theory. They certainly form a good starting point for perusing this volume.

In our opinion the meeting was successful in the sense that it certainly broadened our understanding of what the whole area is about, as well as making most of the participants even more decided than before to devote their scientific efforts to the further development of this well motivated and mathematically very challenging area.

This meeting would not have been possible without the generous help we have received from

Minister für Wissenschaft und Forschung des Landes Nordrhein-Westfalen

Deutsche Forschungsgemeinschaft

Hewlett-Packard, Frankfurt

Mathematischer Beratungs- und Programmierdienst, Dortmund.

We are very grateful for that. We are also very indebted to A.Poigné for helping us so much in the organization of the meeting and editing this volume. Of course we are most grateful to all the participants of the meeting for turning it into a week of very useful scientific and very pleasant personal contacts.

V.Claus,
H.Ehrig,
G.Rozenberg.

Table of contents

INTRODUCTION TO THE ALGEBRAIC THEORY OF GRAPH GRAMMARS (A SURVEY)

Hartmut Ehrig

Technical University Berlin, FB 20

August 1978

ABSTRACT

The aim of this survey is to motivate and introduce the basic constructions and results which have been developed in the algebraic theory of graph grammars up to now. The complete material is illustrated by several examples, especially by applications to a "very small data base system", where consistent states are represented as graphs, operation rules and operations as productions and derivations in a graph grammar respectively. Further applications to recursively defined functions, record handling, compiler techniques and development and evolution in Biology are sketched in the introduction. This survey is divided into the following sections:

1. INTRODUCTION

The algebraic theory of graph grammars is an attempt to describe sequential and parallel graph grammars using graph morphisms and gluing constructions for graphs as basic concepts for the construction of derivations. These gluing constructions are very useful because on one hand they are generalizing the concatenation of strings and on the other hand they are special cases of pushout resp. colimit constructions in categorical algebra so that universal diagram techniques can be used in most of the constructions and proofs. Unfortunately categorical concepts are not known to most researchers in Computer Science and Biology (and there are still many mathematicians not familiar with these constructions!) Actually this had been a disadvantage in making the algebraic approach widely known and we hope that this introductory paper may help to remove this (artificial) burden. Hence we avoid categorical terminology as far as possible in the following sections but the concepts of category theory which are implicitely used are summerized in an appendium.
The aim of this survey is to motivate and introduce the basic constructions and results which have been developed in the algebraic theory of graph grammars. Except of two typical cases we don't give full proofs since they can be found in the literature. Let us point out that this paper is neither a survey on different approaches to graph grammars, for which we refer to /OV.Na 78/, nor a survey on applications. But we will briefly sketch the main fields of applications in 1.1 before summerizing the contents of this survey in 1.2 and giving some technical remarks in 1.3.

1.1 MAIN AREAS OF APPLICATIONS

The development of the algebraic theory of graph grammars was mainly influenced by problems in those areas of applications where "dynamic graph models" can be used (dynamic in the sense of any manipulation of the graph structure).
The main areas of those applications are the following:

- Semantics of recursively defined functions
- Record handling
- Data base systems
- Compiler techniques
- Development and evolution in Biology

In the first three of these areas Church-Rosser properties for graph derivations play an important role. The use of Church-Rosser properties in various areas of Theoretical Computer Science (with correctness of operational semantics being a most "transparent" case) is well-known. Here the data structures to be manipulated are strings or trees. Graph grammar theory allows to extend these applications to arbitrary linked data structures. We will briefly discuss and illustrate the first three of the application areas while the last two are only sketched.

1. Semantics of Recursively Defined Functions

Recursion in programming has often been explicated in terms of macro-expansions of trees. However, efficiency considerations have led to the use of collapsed trees.

Taking for example the recursive definition

f(u,v)=ite(u=0,0,f(u-1,f(u,v)))

we can represent the right hand side by the collapsed tree in Fig. 1.1 instead of a tree which avoids to represent same variables and constants by different nodes.

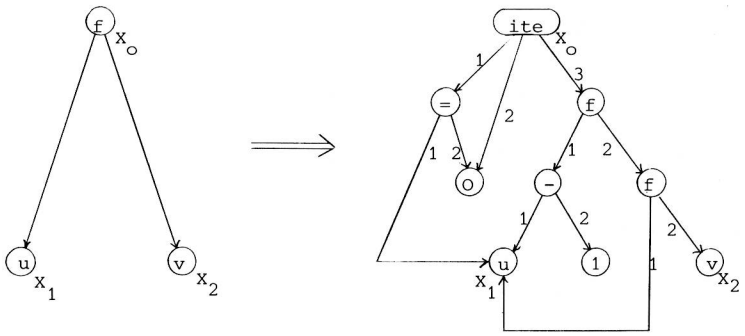


Figure 1.1: Graph Grammar Production for a Recursive Definition

Hence we can express the recursive definition by the graph grammar production in Fig. 1.1 (where the meaning of the gluing items x_0, x_1, x_2 will be given in Sections 2 and 3)

Now expressions, like $f^{\#}(4,f(3,u))+u$, can be represented as "expression graphs" and a macroexpansion of f as an application of the production defined above. Using the Church-Rosser properties for graph derivations we are able to show that all those macroexpansions are satisfying the Church-Rosser property which means that they can be applied in arbitrary order leading to the same result in all cases. Also all or nothing evaluation of a function can be represented as a direct derivation of expression graphs and evaluations and macroexpansions together define a Church-Rosser system. Hence we obtain in /AP.EhRo 76/ a mathematical proof of the following result:

With all or nothing evaluation of given functions, recursively defined functions, represented as expression graphs instead of trees, are single-valued.

An extension of these results including partial evaluation is studied in /AP.Pa 78/.

2. Record Handling

/AP.Eh Ro 77/ discusses a mathematical foundation for reasoning about the correctness and computational complexity of record handling algorithms using algebraic graph theory. A class of pattern matching and replacement rules for graphs is specified, such that applications of rules in the class can readily be programmed as rapid transformations of record structures. When transformations of record structures are formalized as applications of rules to appropriate graphs we can use

Church-Rosser properties to prove that families of transformations are well behaved. In particular, it is shown that any Church-Rosser family of transformations satisfying mild conditions can be combined with housekeeping operations involving indirect pointers and garbage collection without losing the Church-Rosser property. Moreover parallel derivations are used to express the net effect of two transformations as a single transformation. These results and the general theorems that support them can be used to analyze the behavior of a large structure that can be updated asynchronously by several parallel processes or users.

A typical example of a Church-Rosser property involving indirect pointers is given in Fig. 1.2 where in all steps indirection productions are applied (if the target of an arc is colored I the indirection production allows to change the target of this arc to its successor). Let us point out that in Fig. 1.2 starting with the arc colored C we need three while starting with z we only need two applications of the indirection production to end up with the same resulting graph.

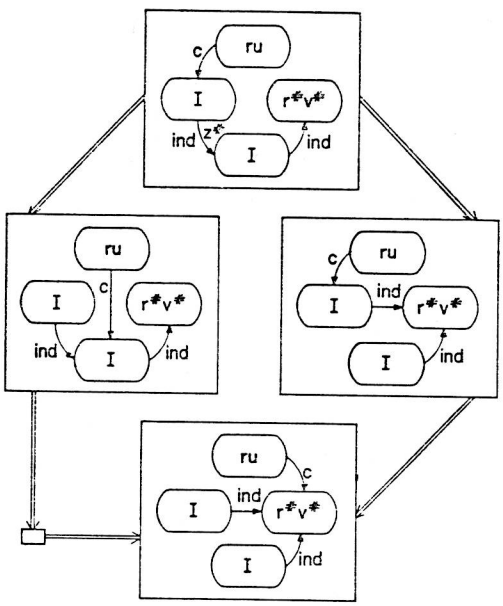


Figure 1.2: Church-Rosser Property of Indirection

3. Data Base Systems

A convenient way to represent a semantic network, the semantic structure of a data base system, is that of using (colored) graphs. In /AP.Eh Kr 76a/,/AP.Kr 78/ and /AP.Eh Kr 78/ it is suggested that manipulations in semantic networks can be expressed using graph productions and derivations. A simple example of a library is given and it is shown how to formalize operations like registering, ordering, lending and returning a book using graph grammar productions and derivations. A typical example for returning a book with catalogue number K 12345 by reader L 0815 is given in Fig. 1.3 (the upper row is the production representing the operation "returning" and the lower row is the direct derivation representing the change of (a small part of) the state of the library system).

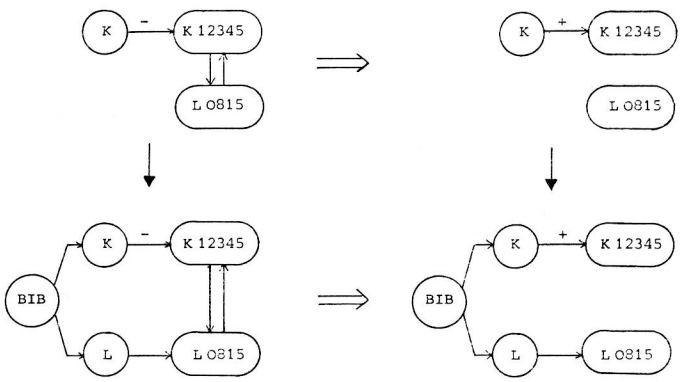


Figure 1.3: Production and Direct Derivation for Returning a Book

More details of this library system will be given in Sections 2 up to 5 where this example is used to illustrate most of the constructions and results in the Algebraic Theory of Graph Grammars. Expecially the operations and consistent states of the library are defined in 3.2 and 3.4.2 as examples of productions and the language of a graph grammar respectively. Using the notion of "independence" for graph derivations (see 4.1) it is shown in Example 4.2 which of the library operations are independent such that they can be applied in arbitrary order or in parallel. Moreover it is discussed in Example 4.8 how the productions ordering and registering (which are not independent if they are applied to the same book) can be combined to a "concurrent production" which has exactly the same effect like sequential application of ordering and registering. The corresponding results (4.4, 4.5, 4.7 and 4.11) in the theory of graph grammars seem to be a basic tool for synchronization problems in data base systems (see /AP.Kr 78/ or /AP.Eh Kr 78/). An extension of some of these results to generalized data base systems is studied in /AP.NePa 77/

and /AP.St 78/.

Since the structure of a semantic network can be defined by a graph grammar the correctness of these structures can be checked using parsing techniques. Two examples of approaches to realize conceptional schemes for data base systems using graph grammars are given in /AP.Go Fu 74/ and /AP.Sc 76/.

4. Compiler Techniques

In several cases compiler techniques can be considered as manipulations of (colored) graphs. We will give two examples: 1. Implementing an incremental compiler, we must consider various lists and a lot of references between them in order to allow insertion, deletion, or substitution of increments. Since the structure is not always a tree it is shown in /AP.Sc 75/ how to establish a syntax-directed concept for implementing incremental compilers using the graph grammar approach for partial graphs in /AP.Sc Eh 76/. 2. For code optimization in a compiler we need program data flow analysis. One of the standard approaches is to construct the control flow graph.

To construct this graph it is standard to group instructions into basic blocks, take these blocks as the nodes, and so on. In /AP.Fa Ke Zu 76/, however, it is shown that most of these graphs can be generated by a graph grammar. A slightly extended version of the algorithm in /AP.Fa Ke Zu 76/ can test in linear time whether a given graph can be generated and construct a parse in the case where the answer is YES.

5. Development and Evolution in Biology

The use of L-systems to model the development of filamental organisms is widely known. The string representation in the mathematical models, however, restrict the application to very simple organisms. In /GL.Lu Li 74/ A. Lindenmayer and K.Culik II have given a mathematical model to extend L-systems to graphs. This allows to model the development of slightly more general (and perhaps also of some higher dimensional) organisms. The main idea is to predict the development of such an organism by calculation of derivation sequences in parallel graph grammars (which are also called graph L-systems). Algebraic approaches to parallel graph grammars are given in /GL.Eh Kr 76/, /GL.Eh Rb 76/, /GL.Ld 78/ and /GL.Eh Ld 78/. In the last two papers the construction of locally star gluing formulas (generalizing locally catenative formulas) is given for graph sequences of (special) parallel graph grammars. This allows to predict the development of organisms by a recursive procedure without calculating the complete derivation sequence.

Finally let us mention that an approach of N. Rashewsky to describe the spezialization and evolution process of organisms was formulated in terms of (sequential) graph grammars in /AP.Eh Ti 75/.

1.2 CONTENTS OF THIS SURVEY AND HISTORICAL REMARKS

Motivated by several other graph grammar approaches (which are reviewed in /OV.Na 78/) the algebraic approach was introduced by H. Ehrig, M. Pfender and H.J. Schneider in

/GG.Eh Pf Sc 73a,b/. Extended versions were given in /GG.Ro 75b/, /AP.Eh Kr 76b/, /GG.Sc Eh 76/ and /GG. Eh Kr Ma Ro Wi 78/ to cover all the applications sketched in 1.1.1 up to 1.1.4. But for this survey we only consider a simplified version of gluing constructions and derivations underlying this approach which is given in Section 2 and 3.

The embedding theorem given in /GG.Eh Pf Sc 73b/ (and improved in /GG.Eh Kr 76/, /GG.Eh 77/, /GG. Kr 77a/ and /GG.Eh Kr Ma Ro Wi 78/ was the first basic result in the Algebraic Theory of Graph Grammars:

Given a derivation sequence $G_o \Longrightarrow \ldots \Longrightarrow G_n$ of graphs and an embedding $h_o : G_o \longrightarrow \bar{G}_o$ then it is possible to obtain an extended sequence $\bar{G}_o \Longrightarrow \ldots \Longrightarrow \bar{G}_n$ using the same productions provided that a suitable JOIN-condition is satisfied. This condition, the embedding theorem (including also the inverse CLIP-construction in 5.3), and some other properties of derivation sequences are discussed in Section 5.

A second basic contribution to the algebraic theory of graph grammars was given by B.K. Rosen in /GG.Ro 75a/, where he started to study Church-Rosser properties of derivations in graph grammars. The results obtained in this and subsequent papers /GG.Eh Kr 75b/, /AP.Eh Ro 76/, /AP.Eh Kr 76a/, /GG.Kr 77b/ and /AP.Eh Ro 78b/ are summarized in the Church-Rosser-, Parallelism-, and Concurrency Theorems in Section 4 (see 4.4, 4.5, 4.7 and 4.11) and in Theorem 5.6 on canonical derivation sequences.

A third basic contribution to the algebraic theory was given by H. Ehrig, H.-J. Kreowski and G. Rozenberg in /GL.Eh Kr 76/ and /GL.Eh Rb 76/ where approaches to parallel graph grammars were given generalizing L-systems from strings to graphs. These algebraic approaches were basically motivated by the original paper /GL.Cu Li 74/ of Culik II and Lindenmayer and by the algebraic approach in the sequential case mentioned above. While some technical results for the algebraic approaches to parallel graph grammars are given in /GL.Eh Kr 76/, /GL.Eh Rb 76/, and /GL.Eh 75/ the first basic result is given in /GL.Ld 78/ resp. /GL.Eh Ld 78/: The construction of a locally catenative formula for dependent PDOL-systems is generalized to the graph case (see 7.4 and 7.6). All this is discussed in Sections 6 and 7. The basic parallel gluing construction, however, is given already in 2.9.

Let us point out that readers which are only interested in parallel graph grammars may skip Sections 3,4 and 5. On the other hand Sections 6 and 7 may be skipped by those which are only interested in the sequential case.

In Section 8 several results concerning graph languages generated by algebraic graph grammars (sequential and parallel case) are summarized with a pumping lemma for type2-graph grammars (proved in /GG.Kr 78/)being of central importance.

As mentioned above there is an appendium in Section 9 on the basic notions of Category Theory which are implicitly used in the other sections. Moreover, the proofs of two results in Section 2 and 4 are given in 9.6 and 9.7 which are typical for the proof techniques used in the Algebraic Theory of Graph Grammars.

References using the first bibliography on graph grammars (prepared by Manfred Nagl) are given in Section 10.

For readers more familiar with the material let us point out that in this survey we only consider productions $p=(B_1 \longleftarrow K \longrightarrow B_2)$ where $K \longrightarrow B_1$ and $K \longrightarrow B_2$ are injective and (colorpreserving) graph morphisms. This is convenient for the presentation but not adequate for some of the applications mentioned in 1.1. On the other hand we are quite sure that most of the constructions and results can be generalized to "structures" (in the sense of /GG.Ra 75/ and /GG.Eh Kr Ma Ro Wi 78/) instead of graphs so that we can cover all these applications (and some more) using only one unified approach.

1.3 TECHNICAL REMARKS

As mentioned above already the reader is not supposed to be familiar with algebraic or categorical notation. All what we need will be carefully motivated and introduced. We only assume basic knowledge of sets and mappings and formal language theory in the string case.

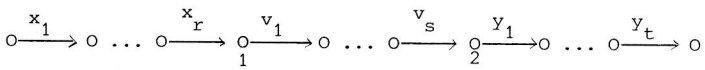
Finally let us remark that (in order to avoid brackets) we sometimes use notations like fx for a map f applied to the argument x, fA for the set $\{fx/x \in A\}$ but not to be confused with gf for the composition of maps applied in the order first f and then g. The notation for graphs and graph morphisms will be introduced in 2.3 and 2.5.

1.4 ACKNOWLEDGEMENTS

Many thanks to Barry K. Rosen, Hans-Jörg Kreowski and Grzegorz Rozenberg for carefully reading of the first draft of this survey, stimulating remarks, discussions and suggestions which led to a considerably improved exposition. Sections 6 and 7 of this survey can be considered as a first part of the paper /GL.Eh Ld 78/ by Axel Liedtke and myself. In fact this is an improved and simplified version of his master thesis /GL.Ld 78/. The bibliography on graph grammars used in this survey was prepared by Manfred Nagl. Last but not least I am thankful to Helga Barnewitz for excellent typing of the paper.

2. GLUING CONSTRUCTIONS FOR GRAPHS

The first major problems to be tackled in building the algebraic theory of graph grammars is to generalize catenation of strings to catenation of graphs, referred to as gluing of graphs. Let us consider a Chomsky production of the form $p=(u,v)$ with nonempty v. Then for each context x,y we obtain a direct derivation $xuy \Longrightarrow xvy$ in such a way that u is replaced by v. More precisely we replace u by v and connect the first symbol v_1 of $v=v_1 \ldots v_s$ with the last symbol x_n of $x=x_1 \ldots x_n$ and the last v_s of v with the first y_1 of $y=y_1 \ldots y_t$. Regarding x,v and y as graphs



we have two "gluing items" 1 and 2 connecting v with the context x and y. (Note that also the case $v=\varepsilon$ (empty word) can be included where the corresponding graph consists of a single node.)

In the general graph case u,v and (x,y) become arbitrary graphs B_1, B_2 and D respectively.

Since the gluing items 1 and 2 in our example are part of v (resp. u) and of the context x and y we will consider in the graph case an "interface" graph K, which is "part" of B_2 (resp. B_1) and D, where B_2 (resp. B_1) and D are disjoint elsewhere. In other words K is exactly the intersection of B_2 (resp. B_1) and D. In this special case we can define the "gluing of B_2 (resp. B_1) and D along K" as the union of B_2 (resp. B_1) and D. Hence the string xuy coresponds to the gluing of B_1 and D along K and xvy to the gluing of B_2 and D along K. This means that a direct derivation in the graph case will consist of two gluing constructions.

For the special case mentioned above where K is exactly the intersection of B and D the gluing of B and D along K is very simple: it is the union of B and D. But this definition of gluing depends essentially on the representations of B, D and K. In general we will have the situation that B, D and K are arbitrary disjoint graphs and that the connection between them is given by graph morphisms $b:K \longrightarrow B$ and $d:K \longrightarrow D$. In this case the gluing G of B and D along K can be defined by the following iterative procedure for G:

```
procedure gluing, parameter (G), input parameters (B, D, K, b, d)
begin G:=B+D        < disjoint union >
for all nodes and arcs k in K do
    G:= identification (G, bk, dk)
end
```

where identification (G, bk, dk) is a procedure which identifies the items bk and dk in the graph G. (Note, that for a more explicit version of the gluing procedure we also have to define graph morphisms $c:D \longrightarrow G$ and $g:B \longrightarrow G$).

Let us consider the following example: