# Computing by Graph Transformation

A Survey and Annotated Bibliography

edited by

H. Ehrig and G. Taentzer

March 1996

# Contents


**1 Introduction** **2**
1.1 This Annotated Bibliography . . . . . 2
1.2 The COMPUGRAPH II Working Group . . . . . 2
1.3 Aims, Paradigms and Research Areas . . . . . 3

**2 Foundations** **4**
2.1 Algebraic Approach to Graph Transformation . . . . . 4
2.2 Context-free Graph Grammars and Graph Transductions . . . . . 7
2.3 Non-context-free Specification Tools . . . . . 11
2.4 2-Structures . . . . . 12
2.5 Graph Rewriting Systems . . . . . 13

**3 Concurrency** **15**
3.1 Theory of parallelism, concurrency, and distribution of Graph Grammars . . . . . 15
3.2 Graph rewriting for specifying concurrent and distributed systems, and relationship with other formalisms . . . . . 18

**4 Graph Transformation for Specification and Programming** **22**
4.1 Term Graph Rewriting . . . . . 22
4.2 Logic Programming and Equation Solving . . . . . 24
4.3 Languages Based on Graph Transformation . . . . . 25
4.4 Application to Language Definitions . . . . . 26
4.5 Various Applications . . . . . 26

**5 References** **29**

# Chapter 1

# Introduction

The theory of graph grammars and graph transformations was originally developed as a common generalization of Chomsky grammars to graphs and term rewriting to graph transformation. Methods, techniques, and results from the area of graph transformations have already been studied and applied in many fields of computer science such as formal language theory, pattern recognition and generation, compiler construction, software engineering, concurrent and distributed systems modeling, database design and theory, logical and functorial programming, AI etc.

This wide applicability is due to the fact that graphs are a very natural way of explaining complex situations on an intuitive level. Hence, they are used in computer science almost everywhere, e.g. as data and control flow diagrams, entity relationship diagrams, Petri nets, visualization of software and hardware architectures, evolution diagrams of nondeterministic processes, SADT diagrams, and many more. Like the "token game" for Petri nets, a graph transformation brings dynamic to all these descriptions, since it can describe the evolution of graphical structures. Therefore, graph transformations become attractive as a "computation paradigm" for complex-structured software and graphical interfaces. In particular, graph rewriting is promising as a comprehensive framework in which the transformation of all these very different structures can be modeled and studied in a uniform way.

## 1.1 This Annotated Bibliography

This annotated bibliography is an attempt to provide an overview of the main past and the most present work on graph transformation. It considers the whole time graph transformation has been studied. The period of the seventies and eighties is described only briefly along few key contributions. More detail is given to the publications in the last six years, the period of the ESPRIT Basic Research Working Group "Computing by Graph Transformation" (COMPUGRAPH I + II). Concerning COMPUGRAPH I these contributions are selected mainly from published papers. All contributions are shortly described. Similarly to the COMPUGRAPH reports the annotated bibliography is structured according to the areas of COMPUGRAPH II.

This annotated bibliography does not give an introduction to graph transformation. For this purpose the reader is referred to [CER79, ENRR87, EKR91, SE94]. A previous bibliography of papers on graph grammars and graph transformation is given in [Nag80].

This annotated bibliography is divided into three main chapters concerning *foundations* of graph transformation, *concurrency* concepts and *graph transformation for specification and programming.*

## 1.2 The COMPUGRAPH II Working Group

Inspired by this observation, some of the most active reseach teams on graph grammars and graph transformation in Europe have established the ESPRIT Basic Research Working Group "Computing by Graph Transformation (COMPUGRAPH)". The first phase of this project, COMPUGRAPH I, was from March 1, 1989, until February 29, 1992. It has been followed by COMPUGRAPH II from October 1, 1992 until March 31, 1996.

The COMPUGRAPH II Working Group consists of the following partners:

- Technical University of Berlin (D), represented by Prof. H. Ehrig
- University of Bordeaux (F), represented by Prof. B. Courcelle
- University of Bremen (D), represented by Prof. H.-J. Kreowski
- University of Brussels/Antwerp (B), represented by Prof. D. Janssens
- University of Leiden (NL), represented by Prof. G. Rozenberg
- University of Pisa (I), represented by Prof. U. Montanari

## 1.3 Aims, Paradigms and Research Areas

Computing originally was done on the level of the "von Neumann Machine" which is based on machine instructions and registers. This kind of low level computing was considerably improved by assembler and high level imperative languages. From the conceptual - but not yet from the efficiency point of view these languages were further improved by functional and logical programming languages. This newer kind of computing is mainly based on term rewriting, which - in the terminology of graphs and graph transformations - can be considered as a concept of tree transformations. Trees, however, don't allow sharing of common substructures, which is one of the main reasons for efficiency problems concerning functional and logical programs. This leads to consider graphs rather than trees as the fundamental structure of computing.

The novelty of our approach is to advocate graph transformations for the whole range of computing. Our concept of *Computing by Graph Transformations* is not limited to programming but includes also specification and implementation by graph transformations, as well as graph algorithms and computational models and computer architectures for graph transformations.

Thus it is supposed to extend current graph grammar techniques in view of specification, programming and implementation concepts. One of the main objections against graphs as fundamental data structure is the high complexity of most graph algorithms (typically NP-complete). Most recently, however, it was shown how to use the theory of graph grammars to reduce the complexity of graph algorithms from intractable (NP-complete) to tractable (polynomial) and even efficient (linear) cases.

On the other hand the efficiency of imperative programming techniques in contrast to functional and logical ones is the reason - as pointed out above - that most of our computers are still based on the von Neumann architecture with its concept of references and pointers. Efficiency of many well-known algorithms is based on that capability. The new idea is to use the well-known concept of references and pointers to allow efficient representation and transformation of quite arbitrary graphs and to overcome the restrictions of tree transformations which are mainly used up to now.

Combining these aspects there is really a good chance for a breakthrough in several areas of computer sciences using graph transformations.

The main research areas of COMPUGRAPH II are

- **Foundations**
- **Concurrency**
- **Graph Transformation in Specification and Programming**

# Chapter 2

# Foundations

The main research goal in this area is a comparative study, unification and the strengthening of the concepts developed in the different approaches, especially the algebraic, algorithmic, logical and the hyperedge replacement approach. This should lead to a much better understanding of common goals and results of different approaches. Moreover, the research on foundations should provide a uniform and solid basis for the investigations in the research areas concurrency and graph transformations for specification and programming.

The *algebraic approach* (section 2.1) developed in the 70's and 80's is based on two pushout constructions in the category of graphs and total graph morphisms. It is now called double-pushout approach, short DPO-approach, in contrast to the single-pushout approach, short SPO-approach, which is based on a single-pushout in the category of graphs and partial graph morphisms.The theory of high-level-replacement systems, short HLR-systems, is a categorical generalization of the DPO-approach. The HLR-approach was also extended to the SPO-approach.

The theory of graph grammars can be seen as an extension to graphs of the theory of formal languages. Two essential aspects are context-free grammars and transformations, called transductions in language theory (section 2.2). Context-free grammars are basic in formal language theory. There are two well-known classes of context-free graph (and hypergraph) grammars: HR grammars, which are based on Hyperedge Replacement, and C-edNCE (or NR) grammars, which are based on Node Replacement (with Neighbourhood Controlled Embedding). Graph transductions can be defined in different ways, by monadic second-order formulas, by tree transducers, by hyperedge replacement grammars.

Non-context-free specification tools are, e.g. hypergraph replacement, monadic second-order logic, graph reduction systems and rewriting of relational structures (section 2.3).

2-Structures form a unifying framework for the investigation of the decomposition and transformation of graph-like structures (section 2.4).

Furthermore, graph rewriting (section 2.5) as a generalization of term rewriting is investigated. Term graph rewriting for the evaluation of functional programs is one of the oldest and most successful application domains of graph transformation. Many properties of term rewriting can be generalized to graph rewriting using the way via term graph rewriting.

Another form of graph rewriting are graph relabelling systems which have been introduced as a formal model for encoding distributed computations on graphs.

## 2.1 Algebraic Approach to Graph Transformation

The algebraic approach to graph grammars is based on categorical constructions, in particular on homomorphisms and pushout constructions. A derivation step of a graph grammar is modeled by a double pushout (DPO) construction or, more recently, by a single pushout (SPO) construction.

## Surveys

In [CMR+96] the state of the art in the DPO approach is given, whereas [EHK+96] deals with the SPO approach. Tutorial introductions can be found in [Ehr87] and [EKL91]. A survey of transformational and algorithmic aspects of the algebraic approach to hypergraph grammars is given in [Hab92b].

## The Double PushOut Approach

In [EPS73] the algebraic approach based on homomorphisms and pushout constructions to graph grammars has been first introduced. For the basic notions see also [Ehr79]. The gluing (given by a double pushout, more details in [EK79]) of arbitrarily labeled graphs is defined allowing a strictly symmetric definition of direct derivations and the embedding of derivations into a common frame.

A two-dimensional hierarchy of graph grammars is given including the classical case of Chomsky grammars and several other graph grammar constructions as special cases. The categorical formulation is extended to arbitrary graphs, like incomplete, topological, stochastic, and multi graphs, in a suitable category in [EK76]. The extension of this algebraic approach to partial graphs, those in which there may be edges without a source or target node, is investigated in [SE76].

Embedding of locally defined derivation sequences into global ones, provided that the start graph is given, is investigated in [Ehr77], yielding two embedding theorems.

Based on the Church-Rosser Theorems in [EK76] canonical derivation sequences are introduced in [Kre77a]. They allow specific, transparent transformations of derivation sequences, which can be used as elementary steps of algorithms acting on derivation sequences. Canonical derivations in graph grammars correspond to leftmost derivations in Chomsky grammars. An extended version can be found in [Kre77], where additionally a pumping lemma for graph grammars is developed.

In [EKMS+81] the double pushout approach is extended to arbitrary relational structures, thus introducing a new approach to the transformation of structures, where the concept of structure is extremely general.

Semantics are dealt with in [Pad82], where transformations of graph-like expressions are called correct if they preserve a given functional semantics of the expressions. Combining the algebraic theory of graph grammars and algebraic specifications it is proven that the correctness of the transformation rules carries over to the correctness of derivations via such rules. Applying this result to LISP it is shown that a LISP interpreter presented by a graph grammar is correct with respect to the functional semantics of graph-like LISP expressions.

## High-Level Replacement Systems

The high-level replacement (HLR) systems, introduced in [EHKP91a, EHKP91b], abstract from concrete categories of graphs to more general categories in such a way that the well-known and desired properties concerning parallelism and concurrency are guaranteed. This yields a categorical framework for different types of replacement systems for complex objects, including graphs, hypergraphs, and structures. Moreover the framework of HLR systems has been applied to the rewriting of algebraic specifications and modular systems in [EP91a, EP91b, EP91c]. Using HLR systems one can clarify the relationship between several choices of specification morphisms in algebraic specifications from the rewriting point of view [EP91b]. The theory of HLR systems (based on the DPO approach) was further extended to include also canonical derivation sequences [EKT94], parallel graph grammars [ET92, Tae96], and embeddings of derivations [PT95]. New applications of HLR systems to transformations of Petri nets and algebraic specifications have been studied in [PER95] and [EP93]. Based on HLR systems, rule-based refinement, its abstract relation to other kinds of refinement, as well as horizontal structuring has been established in [Pad96]. A comparing study between HLR systems in the DPO and the SPO approach was started in [EL93a] and [Wag93].

## The Single PushOut Approach

Single pushout rewriting, as introduced in [LE91], subsumes the classical double pushout rewriting, simplifying the formal treatment: the double pushout consisting of two total graph morphisms is interpreted as one single partial graph morphism from the left- to the right-hand side of the rule. The theory of single pushout rewriting is developed in [Löw90b, Löw90] and published in [Löw93]. It shows that the whole theory for double pushout transformations can be generalized in the new framework. The SPO approach was extended to distributed derivations in [EL93] and to equationally defined graph structures in [Kor92, Kor94]. Attributed graphs were considered in [LKW91, LKW93]. In [CEL+94a] and [CEL+94c] it was shown how to avoid inconsistencies for abstract graph derivations using standard representations of graphs. A new categorical framework of so-called generalized graph structures generalizing comma categories w.r.t. partial morphisms was elaborated in [Kor93a]. A new categorical framework of so-called graph-interpreted graph transformation has been presented in [Kor94a]. Single pushout transformation allowing partial graphs has been introduced in [WG96].

## The Pullback Approach

In [Bau94, Bau95a, Bau95b] a new approach to graph rewriting is introduced based on pullback constructions, which provides a categorical framework for NLC and NCE rewriting. Pullback rewriting is also shown to encompass hyperedge rewriting as well as most cases of double push-out rewriting and to have an intrinsic notion of parallel computation. It appears to be a candidate for a unifying framework of graph rewriting.

## Application Conditions

Graph grammars with application conditions were introduced in [EH85, HHT96]. The concept is general enough to express context conditions like the existence or non-existence of nodes, edges, or certain subgraphs in the given graph as well as embedding restrictions concerning the morphisms from the left-hand side of the production to the given graph. A commutation property and a parallelism theorem for derivations with application conditions are shown in [HHT96]; moreover, context-free graph grammars with application conditions are investigated with respect to their generative power. The embedding properties of graph grammars with application conditions have been studied in [Hec95b]. In [Wag95] the expressive power of graph grammars with different kinds of application conditions are compared. In [HW95] a construction is presented how consistency conditions can be transfered into application conditions, i.e., a given conditional graph grammar is transfered into a conditional graph grammar that satisfies the given consistency conditions. In [HMTW95] a combination of attributed graph transformation with application conditions, amalgamation concepts for rules, and a transaction concept is presented.

## Concurrency

In [Kor94c] minimalization properties of derived rules were used in order to carry over the notion of a concurrent rule from the double to the single pushout approach. In [ELO95] algebraic graph transformations were considered as instances of dynamic abstract data types which were newly proposed as generalizations of the well-known concept of abstract data types. Complementary dynamic abstract data types were used in [Pad95] for giving a semantics to algebraic high-level nets.

## Structuring of Graph Transformations

Little emphasis has been given in the literature to the “in-the-large” structure of graph grammars, i.e., to the study of the relationship among various grammars and of transformations among them. In [CEL+96a] the authors make a first step in this direction, proposing the definition of two categories: the first one has as objects typed graph grammars, while the second one has small categories of graph derivations. Typed graph grammars are a slight generalization of ordinary (algebraic) grammars, where all graphs have a morphism to a fixed type graph. The two categories are shown to be related by an adjunction, which associates each grammar with its free category of derivations. This result generalizes similar results already known for Petri nets. Based on the categories of derivations a categorical semantics for typed graph grammars has been developed. In [CH95] pushouts are used to model composition, and graph grammar morphisms are used to describe refinements of typed graph grammars. Composition and refinement are shown to be compatible with each other and with the proposed semantics.

### Graph Grammar Machines

In [Kre93b], Kreowski introduces the notion of a graph grammar machine. Various translations of computational models into a particular graph grammar machine are constructed. Correctness (in a certain sense) is shown for the translations, leading to several undecidability results for the considered graph grammar approach.

## 2.2 Context-free Graph Grammars and Graph Transductions

The theory of graph grammars can be viewed as an extension to graphs of the theory of formal languages. Two essential aspects are context-free grammars and transformations, called transductions in language theory. There are many types of context-free graph grammars, some based on node replacement and others on (hyper)edge replacement. For the historical developments, see, e.g., [Nag79, Nag80, ER96] and [Hab92a, DHK96], respectively. General requirements for a (graph) grammar to be called context-free are formulated in [Cou87]: the replacement mechanism should be confluent and associative. During the last ten years two main types of context-free graph grammars have emerged: HR grammars, which are based on Hyperedge Replacement, and C-edNCE (or NR) grammars, which are based on (Confluent) Node Replacement (with Neighbourhood Controlled Embedding). NR grammars are more powerful than HR grammars, but HR grammars are easier to understand. Both types of context-free graph languages are robust in the sense that they both admit a number of essentially different characterizations, e.g., by systems of algebraic equations (in the sense of Universal Algebra) [BC87, CER93], by formulas of Monadic Second-Order Logic (MSOL) [Eng91, CE95], and by handle replacement grammars [CER93]. Graph transductions can also be defined in different ways, e.g., by HR grammars [Rao93], by tree transducers [Dre93c], or by MSOL formulas [Cou91, Eng91].

### Surveys

In the monograph [Hab92a] the theory of HR grammars and languages is developed in detail. Among the most important topics are: context-freeness, characterizations of HR languages, a pumping lemma for HR languages, generative power of HR grammars, predicates and numerical functions on hypergraphs compatible with hyperedge replacement.

The survey paper [DHK96] presents the concept of Hyperedge Replacement as one of the most elementary and best studied approaches to (hyper)graph rewriting. As a representative view of the theory of hyperedge replacement, structural properties, decision problems, and the membership problem are considered. For a tutorial on HR grammars, see, e.g., [DK91].

The state of the art of Node Replacement graph grammars (in particular C-edNCE grammars and several of its subclasses) is given in [ER96]. For a tutorial on NR grammars, see, e.g., [ER91].

The survey papers [Cou90a, Cou92a, Cou96a] discuss the use of Monadic Second-Order Logic for expressing graph properties and graph transductions.

A survey on the notion of an MSOL definable graph transduction is given in [Cou94c]. This notion applies to strings, trees, graphs, hypergraphs, relational structures. For graphs it can be viewed as the equivalent of a rational transduction in language theory.

The tutorial [Cou95a] surveys the basic notions of equational and recognizable sets in a Universal Algebra setting. The applications to graphs are immediate for any set of graph operations. For appropriate graph operations, one obtains the notions of HR and NR context-free grammars in terms of systems of algebraic equations. A survey on the use of tree transducer theory, in this Universal Algebra setting, is presented in [Eng94].

## Hyperedge Replacement

In the theory of HR graph grammars and languages essentially three different types of graph properties with nice decidability and structural results have been found: the compatible, recognizable (or finite), and inductive graph properties. In [Cou90b, HKL93] it is shown that these notions are essentially equivalent.

In [Cou90b] it was proved that every MSOL definable set of graphs is recognizable. This innovating paper was the start of a long series of powerful applications of MSOL to context-free graph grammars. In [Cou91] sets of graphs are investigated for which the reverse of the above implication also holds.

In [HKV91] decision problems concerning numerical properties of graphs generated by HR grammars are investigated. Each numerical function on graphs (e.g., the number of connected components) induces a boundedness problem of the form: given an HR grammar, are the function values of all graphs in the generated language bounded by an integer? It is shown that the boundedness problem is decidable if the function is compatible with hyperedge replacement and if it can be expressed inductively by maxima, sums, and products.

It is shown in [CM93] that many functions on graphs that can be defined by MSOL formulas and semi-ring homomorphisms, are compatible with hyperedge replacement (or, inductively computable). Consequently, these functions can be computed efficiently on graphs of bounded tree-width.

It is proved in [CE95] that a set of hypergraphs is HR iff it is the image of a recognizable set of binary trees under an MSOL definable tree-to-graph transduction. This implies that the class of HR graph languages is closed under MSOL definable graph transductions. The analogous results for NR sets of graphs were shown in [Eng91, Cou92c].

The membership problem of HR graph languages is NP-complete in general [RW86]. Polynomial time algorithms for specific cases are described in [Lau90]. In [Vog91] a result by Mac Lane, stating that 2-connected graphs uniquely decompose into components of three different types, is used to obtain a cubic time recognition algorithm for edge-replacement languages consisting of 2-connected graphs. In [Dre93b] the result of Mac Lane is extended to hypergraphs, and the result of [Vog91] is extended to HR languages consisting of $k$-connected $k$-hypergraphs (i.e., hypergraphs with hyperedges of size $k$ only). In [Dre93a] this result is complemented by the proof that HR grammars of order $k$ can generate NP-complete sets of $k$-connected graphs (as opposed to $k$-hypergraphs).

A natural relationship between HR grammars and attribute grammars is established in [EH92].

In [Kus93] a new type of pumping lemma for edge-replacement graph languages is developed, in which the maximum path length of the pumped graph is increased (rather than its size). Using this result, a boundedness problem is shown to be decidable and a sublinear growth property of maximal paths is proved.

In [Mos96a] probabilistic graph grammars are introduced. These are HR grammars equipped with production probabilities. It is shown how to compute statistical measures for fixed production probabilities. Moreover, these probabilities can be determined so as to generate graphs with given statistical information. In [Mos96b] the consistency of such grammars is discussed.

A notion of concatenation of graphs which allows string languages to be interpreted as graph languages is investigated in [EV95, EV96a]; the regular string languages characterize the graph languages generated by linear HR grammars. The string generating power of HR grammars is investigated in [Hab92a, EH91].

## Node Replacement

Important natural subclasses of the NR (or C-edNCE) grammars are the boundary [RW86, ELW90], apex [ELR88, EHL94], and linear [EL89b] NR grammars. Boundary NR grammars are more powerful than HR grammars [Vog93, ER90c]. Many types of NLC (Node Label Controlled) graph grammars [JR80, ER91] are also special cases of the C-edNCE grammars. The results of [HKV91] are generalized to NR grammars in [Eng91, Eng94].

In [Cou95c] a decidable condition is given for a set of graphs generated by an NR graph grammar to be generable by an HR grammar. The condition implies that the NR and HR grammars generate the same class of graph languages of bounded degree. It is shown in [EHL94] that this is exactly the class of graph languages generated by apex NR grammars (or apex HR grammars).

The complexity of the membership problem of NR graph languages was investigated in, e.g., [Bra83, Kau85, RW86, EL89b]. The results concerning graph languages of bounded degree were superseded by those of [Lau90] for HR languages. Other complexity issues are studied, e.g., in [Bra88, SW95, SW96, Wan91, Wan94].

In [Eng89, EO95] it is shown that the NR graph languages can be characterized by regular tree and string languages. For finite string languages the apex graph languages are obtained.

Node replacement in hypergraphs is introduced in [Kle96], yielding hypergraph grammars with neighbourhood controlled embedding (hNCE-grammars). Hyperedge replacement can be simulated by boundary hNCE-grammars. The main result is that confluence is decidable for hNCE-grammars, in triply exponential time.

## Collage Grammars

In [HK91, HKT93] the notion of collage grammar is introduced and investigated. The idea of collage grammar is based on the notion of hyperedge replacement, similar to the case of (hyper)graphs. To generate sets of geometric patterns, they are equipped with hyperedges. As collages represent patterns that may be overlaid, collage grammars provide a syntactic device for the generation of sets of patterns as well as a certain type of fractal patterns (see also [DHKT93]).

The purpose of [DHKT95] is to sketch the basic features of collage grammars as a pattern-generating device based on hyperedge replacement. It is shown that the so-called increasing generalized Sierpinski grammars (gSg) yield self-affine fractals. Taking a certain intersection instead of the overlay of all generated patterns, the same holds for decreasing gSg, and even the converse: essentially, each self-affine fractal can be generated by a decreasing gSg.

In [Tau94] collage grammars are compared with three concepts from fractal geometry that are used to generate fractal pictures, viz. iterated function systems (IFS) with and without condensation, and mutually recursive function systems. It is shown that essentially each of these methods can be simulated by collage grammars.

A first attempt to investigate compatible properties of collages is made in [DK96]. As in the case of HR grammars, decidability results are obtained for compatible properties. In contrast to the situation for graphs, it is difficult to find compatible properties. Slight generalizations of the compatibility concept lead to undecidability. As shown in [Dre96b] there nevertheless exist meaningful classes of context-free collage grammars and IFS for which positive decidability results can be proved, viz. those that use only grid transformations (a restricted type of similarity transformations). It turns out that the language classes have nice closure properties, and that nontrivial questions concerning the generated pictures can be decided. This extends [Dre96a], where it is shown that the connectedness of fractals generated by grid IFS in the plane is decidable.

It is shown in [DHT96] that regular chain-code picture languages can be generated by collage grammars. As a consequence, all known undecidability results for regular chain-code languages can be adapted to picture languages generated by collage grammars.

## Graph Transductions

Using hypergraph replacement instead of tree substitution, the usual top-down tree transducers are generalized to top-down tree-to-graph transducers in [EV94]. Restricting the latter to generate term graphs (i.e., graphs that represent trees) the top-down tree-to-graph-to-tree transducers (top-down tgt transducers) are obtained. It is shown that these have the same power as macro tree transducers. Bottom-up tgt transducers also have the same power [EV96b].

Generalizing the notion of functions compatible with hyperedge replacement (or, the notion of an inductively computable function) the notion of transducibility is defined in [Dre93c]. Roughly speaking, a function $f$ from an algebra $A$ into an algebra $B$ is transducible with respect to a given class of tree transductions if there exists a transduction (in this given class) which, given as input a term that describes an element $a$ of $A$, yields a term over the operations of $B$ whose value is $f(a)$. First results concerning the decidability of boundedness problems (extending the results of [HKV91]) are presented.

In [Dre94] the power of a particular class of transducible functions is investigated, viz. the set of all codings of binary trees as natural numbers that can be computed by top-down tree transducers. It is shown that these functions cannot reduce the size of their input by more than a linear factor if the target algebra contains only monotonic operations.

An investigation of the power of top-down, bottom-up, and tree-to-graph-to-tree transducers (tgt transducers) to compute translations from HR algebras into ER (edge-replacement) algebras is carried out in [Dre95a]. Compositions of top-down and bottom-up tree transducers turn out to be too weak if the operations in the target algebra can define 2-trees or related types of graphs. Tgt transducers are shown to be more powerful. They can compute translations into ER algebras whose operations are so-called 2tree* operations, which are generalizations of the well-known operations to generate 2-trees. By contrast, it is shown in [Dre95b] that top-down tree transducers are as powerful as sharing-free top-down tgt transducers for the computation of functions into a semiring.

In [Dre96c] the notion of computation by tree transductions, and especially its application to graphs and collages, is studied in depth. In addition to the results mentioned above, mainly two new results are presented. The first generalizes the result of [HKV91], in three ways: arbitrary tgt transductions are allowed (whereas compatibility corresponds to deterministic top-down tree transductions), the target domain need not necessarily be the natural numbers with max, $+$, and $\cdot$, and there is no restriction on the input domain at all. The second result says that compositions of top-down tree transducers can compute translations from arbitrary HR algebras into ER algebras containing as operations only (hyper)trees.

## Tree-width and Forbidden Minors

Certain properties of graphs turn out to be intrinsically related to the graph grammar generating mechanism. It is shown in [Lau88, Cou92b] that all HR graph languages are of bounded tree-width; linear HR graph languages are of bounded path-width. In [CE95] it is shown that an NR graph language is of bounded tree-width iff it is an HR language. The relationship with the theory of forbidden minors is investigated in [Cou92b, Cou92a, CP95].

In [BE95] the notion of domino tree-width is introduced and investigated, based on tree-decompositions for which each node of the graph belongs to at most two nodes of the tree. This notion is closely related to apex HR grammars.

By using methods of context-free graph grammars, it is proved in [Lap94] that the tree-width of a planar graph and of its dual differ by at most 1, as conjectured by Robertson and Seymour.

It is proved in [CS95] that one can effectively construct the finite set of forbidden minors for a minor-closed set of graphs given by a HR graph grammar. The corresponding computation method is intractable.

### Context-free Graphs

An infinite graph is context-free (or equational) if it is the solution of a finite set of algebraic equations. They were introduced in [Bau91, Bau92] and investigated in, e.g., [Cou89, Cou90c].

Since every equational graph which is also a Cayley graph, has either 0 or 2 ends, Stallings' theorem implies that the underlying group must be either finite or have a decomposition as an amalgamated product (with finite amalgamated subgroups) or as an HNN-extension (with finite associated subgroups). It is shown in [Sén93] that such a decomposition can be computed in time at most a tower of five exponentials.

Let $G$ be a group such that its Cayley-graph on a finite set of generators is $\kappa$-triangulable. Using geometric considerations, an upper bound on the cardinality of every finite subgroup of $G$ is computed in [Sén96]. As a corollary an upper bound on the accessibility-length of $G$ is obtained. Both bounds are exponential functions of $\kappa$.

In [Pel96] some sufficient conditions are given on the action of the group of automorphisms of a context-free graph, ensuring that this group is finitely generated. In the most general case a geometrical condition, which can be considered as a finiteness condition, is proved.

## 2.3 Non-context-free Specification Tools

### Hyperedge Replacement

In [Kre93a] five variations and extensions of hyperedge replacement (HR) beyond context-freeness are discussed: parallel HR (generalizing L-systems), HR with rendezvous, HR with required presence or absence of context, HR with rule-selection specifications like path expressions, and infinite HR generating potentially infinite graphs.

In [DDK93] a rendezvous mechanism for parallel HR is introduced. This provides the—originally context-free—notion of hyperedge replacement with a restricted kind of context sensitivity. It is shown that even edge replacement with rendezvous exceeds the power of hyperedge replacement without.

### Monadic Second-Order Logic

In general, monadic second-order formulas written with edge set quantifications are more powerful than those restricted to vertex set quantifications. It is proved in [Cou94d] that this is not the case for planar graphs and graphs of bounded degree or tree-width.

It is shown in [Cou95b] that by using MSOL formulas with edge set quantifications, one can define all orientations of a graph or a hypergraph of bounded rank. This is not possible if set quantifications are restricted to vertices. Consequences concern the structure of sets of graphs having a decidable monadic second-order theory.

In [Cou94a, Cou94b, Cou96b] it is investigated whether or not a linear order on the set of vertices of a graph can be defined by MSOL formulas. This is not always possible. Applications concern traces (i.e., graphs representing partially commutative words) and modular decompositions of graphs.

### Graph Reduction Systems

In [ACPS93] graph reduction systems are introduced. It is proved that normal forms can be computed in linear time and this is applied to recognizable sets of graphs of bounded tree-width.

### Models of Graph Transformation

A common framework of graph transformation is discussed in [KR90] where a very general notion of graph grammar is introduced. By fixing various parameters, different well-known classes of graph grammars are obtained.

### Rewriting of Relational Structures

In [EKMS+81] graph rewriting is generalized to the rewriting of general relational strucures. [MSW82] introduces processes that represent derivations of relational structures. In [MSW83] these are used to give semantics to a language for expressing concurrent updating of a relational database. In [MSNT88] specific relational structures are used to give operational semantics to an Algol-like programming language. In [MSW96] Dynamic Graphs are proposed: a model of processes of transforming data, in which branching, concurrency and structures of data can be represented. Data and processes of transforming data are represented by labelled graphs. A particular subclass of dynamic graphs can be interpreted as a semantics of a graph grammar.

## 2.4 2-Structures

2-Structures form a unifying framework for the investigation of the decomposition and transformation of graph-like structures.

### Survey

The state of the art of 2-structures is given in [EHR96].

### Hierarchical Decomposition

It is shown in [ER90a] that the modular decomposition of graphs can be generalized to the clan decomposition of 2-structures in a natural way, using the notion of prime clan of a 2-structure. Thus, each 2-structure can be decomposed into a tree of prime clans, such that the quotient structure of each node and its direct descendants forms one of three types of basic 2-structures: linear, complete, or primitive. This decomposition result is generalized from finite to infinite 2-structures in [HR94].

A Petri net can be synthesized from its state space, modelled as a 2-structure, by the use of regions of that 2-structure [ER90b]. The basic properties of clans and regions of 2-structures, and their relationship, are investigated in [DER94b].

Square systems, introduced in [ER94b], provide a uniform framework for studying decompositions of systems into a hierarchal family of subsets (e.g., prime clans). Such families are related to independent boolean algebra's.

Uniformly non-primitive 2-structures (that have no primitive quotient structures) are studied in [EHPR96]. Several characterizations of these 2-structures (that generalize cographs and transitive VSP digraphs) are presented, e.g., in terms of forbidden subgraphs. It is shown that every MSOL property of uniformly non-primitive 2-structures can be recognized in polynomial time.

A new way of specifying graphs is introduced in [EER96]: through finite sets of strings. The strings of a given (prefix-free) language represent the vertices of the graph, and whether or not there is an edge between the vertices represented by two strings is determined by the pair of symbols at the first position where these strings differ. The main results show an intrinsic relationship between the operations of graph substitution (as used in the decomposition of 2-structures) and language concatenation.

Particular types of 2-structures, called texts, are introduced in [EHPR94]. A text can be viewed as a string with an additional linear ordering on its positions. Context-free text grammars are considered, and it is investigated when a derivation tree of such a grammar corresponds to the decomposition tree of the generated 2-structure.

### Dynamic Labeled 2-Structures

Dynamic labeled 2-structures are introduced and studied in [ER95], as a model of evolution of a network of processors. In a dynamic labeled 2-structure the relationships (i.e., edge labels) between the nodes can change dynamically. The edge labels are naturally taken from a group, and changes are modelled by group multiplication, in order to be able to undo each change. The notion of clan is generalized to dynamic 2-structures, and quotients are investigated in [EHR94a]. In [ER94a] it is shown how nodes can be added to or deleted from (dynamic) 2-structures in a natural way. The group of free invariants of a dynamic 2-structure is studied in [EHR94b]. In [DER94a] dynamic 2-structures are generalized to allow hyperedges; it is proved that communication in a hyperedge is governed by an abelian group.

## 2.5 Graph Rewriting Systems

### Generalized Term Rewriting

In [Plu93] critical pairs are introduced in the DPO approach to hypergraph rewriting. A critical pair lemma is presented which provides a sufficient condition for local confluence of hypergraph rewriting systems (and hence for confluence of terminating systems). It requires that all critical pairs are confluent by derivations satisfying certain conditions. Confluence is shown to be undecidable even for terminating hypergraph rewriting systems. This reveals that a simple characterization of local confluence—as in the case of term rewriting—cannot exist. Critical pair analysis in the framework of SPO graph rewriting is investigated in [LM95].

A necessary and sufficient condition for termination of graph rewriting systems is established in [Plu95]. Termination is equivalent to the finiteness of all forward closures, being certain minimal derivations in which each step depends on previous steps. This characterization differs from corresponding results for term rewriting in that the latter hold only for subclasses of term rewriting systems. When applied to term graph rewriting, the result characterizes termination of arbitrary term rewriting systems under graph rewriting. In particular, it captures non-terminating term rewriting systems that are terminating under graph rewriting.

A concept of graph unification and matching is introduced in [PH96]. The idea is to use hyperedges as graph variables and hyperedge replacement as substitution mechanism. It is shown that a restricted form of graph unification is decidable. The decidability of the general graph unification problem is open. For graph matching, transformation rules are given which compute all solutions to a matching problem. The matching concept suggests a new graph rewriting approach called substitution-based graph rewriting. In this approach, a rule is a pair $(L, R)$ of graphs which is applied to a graph $G$ by (1) choosing a substitution $\sigma$ such that $L\sigma$ is isomorphic to $G$ and (2) applying $\sigma$ to $R$. Substitution-based graph rewriting generalizes the DPO approach in that rewrite steps can copy or delete non-local subgraphs.

In [DS93] the class of so-called constructor equivalent term rewriting systems is defined, for which Thatte's transformation preserves strong sequentiality. A simple characterization of this new class is given and it is shown that it is a subclass of the forward-branching class, and hence of the strongly sequential class. In [DS94] a direct proof of the latter inclusion is given. In [Dur94] a simple characterisation of the forward-branching class is given, together with a quadratic time algorithm that builds a forward-branching index tree.

### Modeling Graph Rewriting via Term Rewriting

In [CM91] an equational algebra is proposed to represent partially abstract graphs, i.e., graphs with both a concrete and an abstract part. Term rewriting over this algebra is shown to correspond faithfully to graph rewriting according to the DPO approach. The resulting term rewriting formalism is proposed in [CMR94] as an original abstract machine, able to describe the evolution of distributed systems. Not only graph rewriting, but also other formalisms like concurrent constraint programming can be faithfully translated into term rewriting systems over that algebra.

### Graph Relabelling Systems

Graph relabelling systems have been introduced as a formal model for encoding distributed computations on graphs. These systems deal with vertex and edge-labeled graphs and satisfy the following constraints: (1) they do not change the underlying structure of the graph but only the labelling of its components, the final labeling being the result of the computation; (2) they are local, i.e., each relabelling step modifies the labelling of a connected subgraph, having a fixed size, of the underlying graph; and (3) they are locally generated, i.e., the aplication of any relabeling rule depends only on the local context of the relabeled subgraph.

In [LM92] a survey is given of classical graph and network algorithms using a tree structure, encoded as graph relabelling systems with priorities (PGRS). The implementation of PGRS is described in [Bil91]. The computational power of noetherian graph relabelling systems with priorities is studied in [LM93]. In particular, the classes of sets of graphs recognized by PGRS are compared with the classes definable by logic formulas.

It is proved in [CM94] that, apart from a few exceptions, minor-closed classes of graphs cannot be recognized by graph relabelling systems. Planar graphs and graphs of bounded tree-width are typical examples of minor-closed classes of graphs.

In [MS94] the well-known problem of election in a tree is considered, and the probability for any vertex to be elected is studied. It is shown that there are one or two vertices with the highest probability of being elected. A simple algorithm to compute these medians is given.

In [LMS95a] six different models of graph relabelling systems are defined and compared; [LMZ95] contributes to understanding the power and limitations of local computations in graphs.

The study of the general problem of characterizing those properties that can be locally computed on a graph by graph relabelling systems, is initiated in [LMS95b]. It is proved that graph relabelling systems can recognize all graphs in which a given label has majority, but no such system can recognize all graphs in which a given label has majority with a prescribed number of exceeding occurrences.

Applications of the Kronecker product to local computations in graphs are studied in [BM96].

# Chapter 3

# Concurrency

Formalisms based on graph rewriting may be used in a quite natural way to specify concurrent and/or distributed systems. In fact, distributed states are naturally represented by graphs, and graph transformation rules can be used to model local changes in a distribute state. The area of concurrency in the theory of graph grammars can be roughly divided in two main streams of investigation (which clearly have strong interconnections):

1. The first line of investigation concerns the analysis of the various aspects of parallelism, concurrency and distribution of graph grammars and of their derivations. This area includes the study of parallel and distributed grammars; the analysis of properties of causal dependency or indepencence among graph derivation steps; the definition of suitable equivalences among graph derivations, and of canonical representatives for such equivalence class (as canonical derivations, processes, and concurrent derivations); the study of truly concurrent semantics for graph grammars. Such topics have been investigated for various rewriting formalisms, including for example the DPO and SPO approaches to graph grammars, HLR systems, and transition systems with parallelism structure.

2. The second line of research aims at investigating in a more direct way the suitability of graph grammars for the specification of concurrent and distributed systems. This is done either by defining specific formalisms based on graph rewriting that explicitly address such modelization issues, as in the case of ESM (Extended Structure Morphsm) systems and GDS (Grammars for Distributed Systems), or by studying the relationship between graph grammars and other formalisms suitable for specifying concurrent and distributed system. In most cases the basic aspects of such formalisms are translated into corresponding notions for graph grammars, and it is proven that the resulting grammar faithfully model the operational behaviour of the original system. Such a translation sometimes provide useful hints for some cross-fertilization between the theory of graph grammars and the theory of other formalism, as it happened for Petri nets. Other formalisms considered include Actor Systems, Concurrent Constraint Programming, Asynchronous processes, the CHARM, and Statecharts.

In the following subsections we list and shortly describe the various topics under which the main contributions to the area of concurrency produced before and during the two COMPUGRAPH projects can be classified.

## 3.1 Theory of parallelism, concurrency, and distribution of Graph Grammars

### Parallelism, Concurrency and Distribution

Parallelism has been investigated from the beginning of the algebraic approach to graph grammars

Parallel graph grammars [EK76a, EK76b] have been introduced to formalize parallel rewriting in multidimensional structures which can be represented as labeled graphs. On the one hand this concept generalizes Lindenmayer-systems, usually defined for strings, to labeled graphs, yet on the other hand this is the parallel version of the algebraic approach to graph grammars as given in [EPS73]. A comparison to other approaches to parallel graph grammars can be found in [ER76].

The concurrent manipulation is dealt with in [ER78]. Given a sequence of manipulation rules together with dependence relations one can construct a single concurrent manipulation rule with the following property: Each application of the sequence of rules to a multidimensional information structure – such that the dependence relations are respected – can be performed in a single step applying the concurrent manipulation rule to the same structure. Moreover this becomes a bijective correspondence between such manipulation sequences and concurrent manipulations. This Concurrency Theorem is formulated and proved in the framework of the algebraic approach to graph grammars. [ER80] is concerened graph manipulations and with the Church-Rosser and the Parallelism Theorem based on the notions of parallel and sequential independence.

[EHR86] provides a common framework to study transformations of structures ranging from all kinds of graphs to relational data structures. Transformations of structures can be used as derivations of graphs in the sense of graph grammars, as updates of relations in the sense of relational data base, or even as operations on data structures in the sense of abstract data types. Parallel and concurrent transformations are constructed from given sequential ones and the sequentializability properties of complex transformations are studied.

In [PPEM87] the algebraic approach has been extended by imposing a simple structure on the sets of labels to allow variables both in graphs and productions. Instantiations are performed by graph homomorphisms. Using relative unification, the authors define the composition of rules and prove the Concurrency Theorem in this more general framework. By restricting the attention to rooted, directed, acyclic graphs standard term rewriting with first order substitutions can be represented.

In [BEHL87] concepts for the distribution of state graphs are discussed within the framework of the algebraic approach to graph grammars. First ideas are presented concerning distributed derivations and distributed graph grammars. Parallesism of graph transformations for distributed state graphs are studied in [EBHL88]. The distributed parallelism theorem provides a bijective correspondence between distributed local derivations and parallel derivations of corresponding global states. This result holds for strict interface preserving derivations while in general distributed derivations are not reducible to global state derivations.

Classical results concering parallelisms and concurrency have been generalized to high-level replacement systems [EHKP91a, EHKP91b, ET92]. They are formulated in an axiomatic framework based on categories and double pushouts. The basic notions, like productions, parallel and sequential independence, are introduced for those systems and lead to Church-Rosser, Parallelism and Concurrency Theorems. Specific conditions, called HLR-conditions, have been assumed in order to obtain the above results. Several examples of high-level replacement systems are discussed and classified with respect to these conditions.

Parallel replacement on graphs and high-level structures is defined on the DPO-approach in [Tae96]. Different types of replacement are compared with each other and also with sequential graph grammars. Parallel graph grammars are also considered in [Bet92]. Distributed SPO-derivations are studied in [EL93].

In [Tae96b] synchronous and asynchronous graph transformations are considered on the basis of distributed transformations in the SPO-approach. Some criteria are given there preserve synchrony according to the definition proposed. The distributed graph transformations considered in that work are allowed to split a global graph into local parts once. If this splitting can be iterated we arrive at hierarchically distributed graph transformations in [Tae95]. This kind of transformation offers the possibility to transform the distribution structure as well as the local parts in a rule based way.

In [TS91] massive synchronized parallel behaviour of atomic actions is used to model the movement of complex objects. The model is based on a generalization of the algebraic gluing condition and, by a two-level transformation, it is able to describe linear transformations, rotation, and the interaction of several complex objects.

For hyperedge replacement grammars, a mode of parallel rewriting as known from Lindenmayer systems is introduced and studied in [Kre92].

A new synchronization operation on productions is introduced in [CR95, CR96], which generalizes the parallel composition of productions in the DPO approach. Such operation is applicable even if the two productions are not parallel independent (provided that they do not delete the same item): In such a situation, the application of the two productions may not be sequentialized in some order.

## Amalgamation

The generalization of the Parallelism Theorem yields the Amalgamation Theorem in [BFH87]. This theorem drops the assumption of parallel independence. For each pair of productions together with a relational production a single amalgamated production can be constructed. The Amalgamation Theorem states that graph derivations that respect the given associations can be amalgamated to a single derivation via the amalgamated production. The amalgamation mechanism can be used to handle synchronization phenemena.

Amalgamation and distributed derivations have been studied for the single pushout approach [Löw93, EL92c]. Amalgamation in the single pushout formulation can easily be connected to asynchronous and distributed behaviour (see [Löw90] for theory and examples). Especially, applications of the single pushout approach to the modelling of distributed systems have been considered in [EL92c, EL93]. It is proved that synchronous and asynchronous parallel rewriting do not coincide in this setting. This is due to the fact that specific synchronization problems of distributed systems are faithfully reflected in the graph model.

## Truly concurrent semantics for graph grammars

In a joint research activity started in 1992, the Berlin and Pisa groups of COMPUGRAPH have provided graph grammars in the DPO approach with a truly concurrent semantics in terms of Event Structures. In order to define such a semantics in a satisfactory way, in [CEL+94c] the authors addressed the problem of formalizing what is precisely meant by reasoning about graphs and graph derivations "up to isomorphism": they showed that some incongruences arising from a naive use of isomorphisms in the definition of "abstract" graphs and graph derivation can be solved by restricting the class of allowed isomorphisms to a proper subclass of so-called "standard isomorphisms". This result has been exploited in [CEL+94a], where the main result is the definition of an original equivalence on graph derivations which is compatible with sequential composition and guarantees the uniqueness of canonical derivations. This equivalence allows for the definition of a new category having abstract graphs as objects and abstract derivations as arrows.

In [CEL+94b] an abstract event structure semantics has been given to so-called safe graph grammars following the double pushout approach. The restriction to safe grammars allowed the authors to get rid easily of the problems (stressed in [CEL+94a]) arising from the existence of non-trivial automorphisms. In [CEL+96b] the event structure semantics has been extended to arbitrary, consuming graph grammars, fully exploiting the techniques developed in [CEL+94a].

Motivated by a corresponding application to actor systems, an event structure semantics is also proposed in [Kor94b] for the single pushout approach. This kind of semantics is based on concurrent derivations which can be considered as graph grammar processes ([KR95a]).

## Canonical Derivations

Canonical derivations, i.e., derivations where each production is applied "as early as possible" (as soon as all its preconditions are generated), have been investigated for the first time in the DPO approach [Kre77]. In the graph grammar theory they are known as the counterpart to leftmost derivations in Chomsky grammars. As such, they can be regarded as standard representatives for equivalence classes of derivations with respect to the so-called "shift-equivalence". Such an equivalence relates derivations which apply non-causally related derivation steps in any order.

In [EKT94], Ehrig, Kreowski and Taentzer generalize canonical derivations, previously studied for string and graph grammars only, from graph grammars to high-level replacement systems, short HLR-systems. It turns out that basic results concerning synthesis and analysis of parallel derivation sequences in HLR-systems, obtained in previous papers, can be extended to construct canonical parallel derivation sequences which are optimal w.r.t. leftmost parallelism. The main results show the existence and uniqueness of canonical derivations under weak assumptions for the underlying categories of HLR-systems. These results are specialized to graphs, hypergraphs, Petri nets, algebraic specifications and others by classifying the underlying categories with respect to the assumptions. This leads to interesting new results for most of the corresponding HLR-systems. Furthermore, a direct approach to obtain canonical derivation sequences in the single pushout approach was elaborated in [LD94].

Kreowski's paper [Kre94] presents an axiomatic approach to canonical derivations in the framework of transition systems with parallelism structures. Canonical derivations are studied in this paper as candidates to represent the processes in a concurrent system or, more exactly, as representatives of equivalent views on the processes. The main results give sufficient conditions for existence and uniqueness of canonical derivations.

A different kind of representative for shift-equivalent classes of derivations has been proposed in [CMR96] and [Kor94b], for the DPO and the SPO, respectively. In both approaches, such a representative is obtained as a colimit of a derivation, regarded as a diagram in the category of Graphs: although such colimit object is not, in general, a derivation (unlike canonical derivations), the causal ordering of direct derivations can be extracted in an easier way.

## 3.2 Graph rewriting for specifying concurrent and distributed systems, and relationship with other formalisms

### Petri Nets

A graph grammar is constructed from a given Petri net in [Kre81]. There direct derivation steps correspond to firings of transitions in the net. This close relationship between Petri nets and graph grammars allows for the comparison between the underlying concepts of concurrency of both theories.

In [CEL$^+$94b] the authors study the relationship between graph grammars and Petri nets, which are regarded as grammars acting on discrete graphs and where the interface of each transition is empty. Many papers explore further this relationship, applying to graph grammars well-known constructions and techniques already developed for nets. For example, in [CMR96] "graph processes" are introduced, which directly generalize Golz-Reisig net processes. Similarly, the adjunction relating the category of typed grammars and the category of typed graph derivations introduced in [CEL$^+$96a] generalizes to grammars a similar adjunction proposed for nets in [MM90]. [CM95, Cor95] contain a summary of some categorical techniques used in literature for providing Petri nets with a truly concurrent semantics in terms of event structures, and a description of the state of the art of the research aimed at generalizing such semantics to graph grammars.

Algebraic high-level nets were revisited in [EPR94] and extended to include transformations in [PER95] and initial markings in [RP94]. The relationship between algebraic high-level nets and attributed graph grammars is studied in [KR94, KR95b].

### GDS: Grammars for Distributed Systems

Grammars for Distributed Systems (GDS) were introduced in [CM83, DM87] as a specification formalism, based on graph rewriting, for concurrent and distributed systems. The state of a GDS is a graph representing a distributed system made of processes (modelled by hyperedges) connected through ports (nodes), as well as the history of events that led to the current configuration. The evolution of processes is specified by context-free graph rewriting rules, which may offer certain actions on ports. Such rules cannot be applied autonomously: a rule offering an action to a port must be synchronized with a rule for the other process connected to that port offering the same action. Thus the evolution of the system is obtained by the synchronized firing of many rules, producing an event (that is added to the history of the system), and the new configuration. This synchronization mechanism can be considered as a form of rule amalgamation *ante litteram*.

## Actor Systems

"Actor grammars" have been introduced in [JR89] as a formal model for actor systems. This formalism has been generalized to a more powerful graph rewriting formalism, called ESM (Extended Structure Morphism) systems, described in the next subsection.

In [Kor93] actor systems were modelled by a distinguished algebraic graph grammar following the single pushout approach in the sense of [Löw93, LKW93]. THE idea was to reuse the available theory for graph grammars in [Löw93], attributed with algebras in the sense of [LKW93]. Moreover, in [Kor93], faithfullness of the model is shown w.r.t. a suitable system of axioms. A number of consistency requirements for actor systems could advantageously be described by conditional equations. The corresponding extension of SPO-transformations and its theoretical implications are carefully presented in [Kor92]. Its application to actor systems is contained in [Kor94], showing the use the effect of certain equationally defined constraints.

In [Kor95] a true concurrency semantics particularly adressing the fairness requirement of finite delay is given to an algebraic graph grammar model of actor systems.

## ESM systems

Research on ESM (extended structure morphism) systems originates from work on a more restricted type of graph grammars, called Actor Grammars. In the framework of COMPUGRAPH I, these were introduced as a graph rewriting model of actor systems. ESM sytems arise from actor grammars by lifting a number of restrictions on the form of productions which, although reasonable if one is interested in modeling events in an actor system, are rather artificial from the viewpoint of graph rewriting. Moreover, one often wants the possibility to express the effect of a whole computation by a single production, forgetting about the way the computation is built from atomic steps; it turns out that this also requires a notion of production that is more general than that used in actor grammars.

Graph rewriting in ESM systems is rather fundamentally different from graph rewriting in the "algebraic" approach because it is based on the use of a nontrivial embedding mechanism: if a graph $g$ is rewritten, resulting in a graph $h$, then $h$ inherits edges from $g$ via this embedding mechanism. There is no condition excluding "dangling edges"; whenever a production $\pi$ can be applied to $g$, then $\pi$ can also be applied to any graph obtained from $g$ by adding edges to it. This is one of the reasons why we believe the theory of ESM rewriting may be quite different from that of algebraic graph grammars, and why it is worthwile to continue its investigation in spite of the fact that the theory of algebraic graph grammars is now much more developed. The research on ESM rewriting in the framework of COMPUGRAPH II was mainly focused on the development of a process semantics, and of structuring methods for ESM systems that should be compatible with this semantics.

In order to shed some light on the relationship between algebraic and ESM graph rewriting, it has been shown in [Jan94] that an atomic rewrite in an ESM system corresponds to a pushout construction in a category of graph structures with arrows that extend the usual structure morphisms. These arrows are called extended structure morphisms, and they are very similar to the structured transformations from [JR89]. The name ESM system is motivated by the fact that productions are in fact ESM morphisms.

In [Jan94] the notion of a computation structure is introduced; a computation structure formalizes a rewriting process of an ESM system. The notion may be seen as an improved and generalized version of the computation graphs from [JLR93]. It is shown in [JM95] that abstract computation structures and concrete configuration graphs form a category that is a natural generalization of the well-known notion of a transition system. The set (or category) of computation structures may be taken as the basic process semantics of an ESM system.

In [JM95] various mechanisms are presented to extract, from the set of computation structures, other semantics that are more abstract in the sense that the corresponding semantic equivalence identifies more systems. One interesting possibility is to extract from each computation structure its external effect, described by the substructure induced by the sets of minimal (input) and maximal (output) nodes; the so obtained structures may be used as productions in a higher level description of the system. The operation is to an extent similar to, e.g., hiding in CCS. In [JLR93] it is shown that such an operation can indeed be defined; in [JM95] it is shown that it behaves well with respect to the composition of computation structures. Other possible mechanisms are (1) restricting the global system states, and (2) restricting the visibility of the local states. Both are discussed in [JM95]. It is also shown that the various abstraction mechanisms presented are compatible with the sequential composition of computation structures (i.e., the abstraction of the composed structure is the composition of the abstractions).

The important issue of system composition and decomposition is considered in [Jan94], [JM95] and [Jan95]. Whereas system composition is formalized as union(of sets of productions), composition of the process semantics is based on the composition of individual processes: computation structures may be glued together over some of their input and output nodes, yielding a computation structure describing their combined rewriting process. Sequential composition is the special case where the gluing occurs only over output nodes of the first and input nodes of the second computation structure. It is shown in [Jan95] that a refined version of the semantics, where computation structures are equipped with additional information about the context in which they may appear, is compositional.

## Concurrent Constraint Programming and Contextual Nets

The use of graph grammars as a computational model for concurrent constraint programs, studied by Montanari and Rossi in [MR93], allowed the authors to stress the importance of the context-dependent nature of graph rewriting in the algebraic approach. In fact, graph rewriting in this approach is context dependent, where the context is the "interface graph", that is, what is needed for a production to take place, but is not affected. This turns out to be useful for all computational models based on shared memory (to model read operations), and also for generic concurrent system (since it allows one to express more concurrency, minimizing the causal dependencies). In [MR95] the authors added this contex-dependent aspect to Petri nets, which are usually used to model resource-based concurrent systems, extending them with context conditions, and obtaining the so-called *contextual nets*. They propose process semantics of such new nets, which are a conservative extension of the classical case. Such contextual nets are used to model concurrent constraint programs in [MR94]. In this way, they retain all the convenient features of graph grammars, while adding the possibility of representing all computations in one unique structure. This allows to show both the concurrency and the choice points present in the execution of a (possibly nondeterministic) program.

## CHARM

In [CMR94] the authors propose an abstract machine called CHARM, based on term rewriting in a suitable algebra. The states of such a machine have both a local (hidden) and a global (visible) part, and are closed under parallel composition. The tranisitions (rewrite rules) are required to preserve the global part. This constraint allows one to recover in the framework of term rewriting systems the essential features of context-dependent rewriting formalisms. It is shown that both graph grammars and concurrent constraint programming can be faithfully implemented on this abstract machine.

### Statecharts

The issue of modeling the operational behaviour of specification formalism Statecharts via suitable graph grammars has been addressed by Maggiolo-Schettini and Peron in a number of papers, starting with [MSP92] where a compositional semantics of Statecharts based on graph rewriting is proposed.

In [MSP94] graph rewriting is proposed as a proper "abstract machine" for defining an operational semantics for Statecharts. A statechart is translated into a set of graph productions and a step from a configuration of a statechart to another is described in terms of a derivation from a graph describing the start configuration to a graph describing the reached configuration. The sequential behaviour of the represented statechart is described by the derivation graph. In the full version of the paper, [MSP93], it is shown also that a more abstract partial order semantics of Statecharts can be obtained by suitably mapping partial orders over transitions onto the derivation graph.

In [MSP96] a new translation has been given that supports different semantics by proper selection of derivations. In particular a new semantics is supported that allows to view statecharts as a formalism to describe dynamically evolving systems. Also a partial order semantics can be easily obtained.

# Chapter 4

# Graph Transformation for Specification and Programming

Before COMPUGRAPH II executability of algebraic specifications using graph transformations was one of the main aims. It turned out that jungle evaluation, studied by the group in Bremen, and term graph rewriting, investigated in the ESPRIT Basic Research Action SEMAGRAPH, were closely related to each other and both are very useful for efficient execution of algebraic specifications on one hand and functional programs on the other hand. In the second phase of COMPUGRAPH the aim is even broader: We want to justify the new paradigm that computing by graph transformation is a fundamental concept for specification and programming on different levels of abstraction in the software development process. This includes in addition to term graph rewriting the integration of graph transformations, graph rewriting languages for specification of programs and software systems modularity concepts for graph transformation systems as well as tools and applications.

Functional expressions and logical formulas represented by graphs and hypergraphs allow sharing of common subexpressions or subformulas and hence an efficient evaluation using graph and hypergraph rewriting (section 4.1). Hyperedge Replacement Jungle Rewriting is a formalism well suited for the modeling of logic programming and term rewriting systems (section 4.2).

several specification languages based on graph transformation have been proposed. One of the most famous and very far developed is PROGRES which has already been successfully used for pratical applications. All currently existing specification languages based on graph grammars lack a module concept which enables to structure large specifications. Such a concept is heavily needed to master the complexity of specifications of real-world applications. First approaches in this direction are reported in section 4.3. Furthermore GRACE, a new graph and rule centered language in an early stage of development involving researchers from Aachen, Berlin, Bremen and Leiden, is presented. It is intended as a specification and programming language for rule-based systems employing graph rewriting. The dream behind GRACE is to make the potentials of graph grammars and graph transformation systems more visible and applications of them more popular.

Moreover, graph transformation is applied to languages definitions, e.g. to model the object-based database language Troll Light, to specify hybrid query languages for databases or to define the syntax of visual languages (see section 4.4).

In section 4.5 various applications of graph transformation are listed.

## 4.1 Term Graph Rewriting

Functional expressions can be represented by graphs and hypergraphs, making it possible to share common subexpressions. The evaluation of expressions can then be realized by graph and hypergraph rewriting. This observation has led to one of the best developed applications of graph transformation techniques to the area of specification and programming.

In [Plu93a], Plump studies rewriting of term graphs—hypergraphs representing functional expressions such that common subexpressions can be shared—based on the *jungle evaluation* approach which was developed in [HP88, HP91, HKP91]. Rewrite steps with term graphs include applications of term rewrite rules as well as "folding steps" which identify common subexpressions. Different aspects of this model of computation are considered: (1) It is shown that term graph rewriting is complete with respect to equational validity in the same sense as term rewriting is. (2) Term graph rewriting and term rewriting are compared with respect to confluence. (3) Termination and convergence of term graph rewriting turn out to be modular properties, in contrast to the situation for term rewriting. The modularity of termination was first shown in [Plu91a].

In general, a term may be not normalizable by term graph rewriting although a normal form (according to term rewriting) exists. Term rewriting systems for which every term can be normalized by graph rewriting are called *graph-reducible* in [Plu91b]. It is shown that the following property is sufficient for graph-reducibility: every term having a normal form can be normalized by parallel term rewrite steps in which a rule is applied to *all* occurrences of some subterm. As a consequence, a broad class of term rewriting systems which includes all terminating and all orthogonal systems can be shown to be graph-reducible.

In his dissertation [Plu93b], Plump develops a theory for the evaluation of functional expressions by hypergraph rewriting, based on the double pushout approach. The main results concern the following topics: soundness and completeness w.r.t. equational validity; the relationship to term rewriting w.r.t. termination, confluence and related properties; sufficient conditions for confluence based on critical pairs; criteria for the preservation of termination and convergence under combinations of systems.

It is well-known that the equational theory of an equational specification is decidable if the specification can be presented as a confluent and terminating term rewriting system. The corresponding fact holds for term graph rewriting with folding (or "collapsing"), and here the decision procedure applies to more systems than in the case of term rewriting. In order to provide a test for confluence, Plump [Plu94] introduces *critical pairs* in term graph rewriting and establishes a Critical Pair Lemma as an analogue to the well-known result in term rewriting. This leads to a decision procedure for confluence in the presence of termination. As a by-product, the procedure can be used as a confluence test for term rewriting and as such it extends the classical test of Knuth and Bendix because it applies to all terminating and to certain non-terminating term rewriting systems.

Hoffmann [Hof94] studies the extension of jungle evaluation by *memoïzation*, an implementation technique known from functional programming languages: by rather simple modifications of the rules, the results of function applications are stored, so that they can be reused later on if the same function applications occur again. Under some conditions assuring that no cycles are introduced, jungle evaluation with memoïzation computes correct normal forms if simple jungle evaluation is confluent; if the evaluation rules are non-overlapping, the intermediate graphs are also correct. Within this framework, strategies for *full* and *lazy* memoïzation can be realized. This result does not depend on a particular strategy for applying evaluation rules. The approach has been extended to terminating and convergent systems, and to cyclic term graphs for orthogonal rewrite systems.

The theory of term graph rewriting has been extended in the last years to the case of graphs with cycles, which can represent infinite, *rational* terms, and it has been shown that rewriting of these "term graphs" corresponds to infinite term rewriting sequences [KKSdV90]. In [CD96], Corradini and Drewes take a different point of view, showing that if the rewriting of term graphs is performed using the double pushout approach, then a single graph rewriting can correspond to an *infinite parallel rewriting*, i.e., to the application of a rewrite rule to infinitely many redexes in parallel. The theory of infinite parallel rewriting has been developed in [Cor93], fully exploiting the algebraic structure of the complete partial order $CT_\Sigma$, containing all possibly partial, possibly infinite terms over a signature $\Sigma$.

Functional term graph rewriting systems provide a computational model for functional programming languages. Functional programs, and hence functional term graph rewriting systems, are suited as executable specifications. The use of functional term graph rewriting systems as a specification language was investigated by Koopman, Smetsers, van Eekelen, and Plasmeijer [KSEP93, KEP95]. To increase the expressive power of such a language, Koopman [Koo94] considers the notion of subtyping with inheritance. In order to speed-up the evaluation, Zweije and Koopman [ZK93, Koo93, KZ94] transform term graph rewriting systems into semantically equivalent systems that are operationally more efficient.

Kuske [Kus95b] shows how to implement the $\beta$-reduction of the $\lambda$-calculus by hypergraph rewriting according to the double-pushout approach. The resulting model, called *collapsed $\lambda$-tree rewriting*, is shown to be sound with respect to $\beta$-reduction and complete with respect to the Gross-Knuth strategy (which reduces all redexes in a $\lambda$-term in parallel). As a consequence, a collapsed $\lambda$-tree is normalizable by collapsed $\lambda$-tree rewriting if and only if the represented term is normalizable by $\beta$-reduction.

## 4.2 Logic Programming and Equation Solving

Based on a previous work with Montanari, Ehrig and Löwe [CMR$^+$91], Corradini and Rossi propose in [CR93] *Hyperedge Replacement Jungle Rewriting* as a formalism suited for the modeling of Logic Programming and Term Rewriting Systems. In fact, by exploiting the fact that pushouts in the category of jungles correspond to term unification, they propose a suitable representation of definite clauses by graph productions, and they prove that a direct derivation based on the standard double-pushout construction faithfully corresponds to a resolution step. A similar encoding is proposed for the rules of a term rewriting system as well: the advantage of such an encoding with respect to other proposals in the literature is that non-left-linear rules can be handled in a completely uniform way.

The relationship between Logic Programming and graph grammars is explored also in [CRPP91], where a new representation of program clauses as jungle productions is proposed. A resolution step is modeled there by an algebraic construction involving three pushouts. The advantage of such approach is that it matches the logic programming spirit better, since it clearly separates the unification and rewriting phases of each resolution step. Furthermore, classical results on concurrency for graph grammars have been shown to hold for logic programming as well, and to provide a precise algebraic characterization of specialization and unfolding of clauses.

In [CW94], Corradini and Wolz propose the use of Hyperedge Replacement Jungle Rewriting (as introduced in [CR93]) to model, at three different levels of abstraction, the behaviour of an abstract machine for lazy narrowing. The Narrowing Calculus consists essentially of term rewriting with unification and subsumes both Conditional Term Rewriting and Logic Programming. It allows to compute solutions to equations over (conditional) equational specifications. The advantage of modelling narrowing via graph rewriting is that the sharing of common subterms, which is exploited by most of the narrowing implementations, can be modelled directly. Each level of abstraction corresponds to a specific graph rewriting system, together with some application conditions: the highest level is shown to correspond almost one-to-one to narrowing, while the lowest one is shown to be close to the actual implementation. The three descriptions are related by formal results, which show their consistency under certain assumptions. The whole framework provides powerful formal tools for the study of correctness of narrowing implementations.

In [HP95], Habel and Plump apply their recently introduced concept of graph substitution to term graphs. This yields a uniform framework for unification, rewriting, and narrowing on term graphs. The notion of substitution allows definitions of these concepts that are close to the corresponding definitions in the term world. The rewriting model obtained is complete for equational deduction. For term graph narrowing, a completeness result is established which corresponds to Hullot's classical result for term narrowing: if term graph rewriting is convergent, then narrowing can find for every solution of a given equation a more general solution. The general motivation for using term graphs instead of terms is to improve efficiency: sharing common subterms saves space and avoids the repetition of computations.

## 4.3 Languages Based on Graph Transformation

A formalism for rewriting hypergraphs in a controlled manner has been developed by Maggiolo-Schettini and Winkowski in a series of papers, including [MSW91, WMS92, MSW92, MSW94, MSW95]. The formalism is essentially a programming language with rewriting rules playing the role of basic instructions. The programs are built from productions by means of constructors which specify order and modalities of the rewriting steps. When applied to a hypergraph such programs produce transformations of the hypergraph. Productions and programs may contain parameters to point to specific elements of the hypergraphs and can be used as a mechanism to pass data from one rewriting step to the other.

The language has been endowed with a structural operational semantics and with a denotational semantics which assigns with each program a correspondence between states of data and execution trees. The language may be suitable for various purposes. It may be thought of as an intermediate language with high level feature for programmanig execution of logic programs, or as a language for writing programs which may serve as easy verifiable prototypes of hardware or software systems.

All currently existing specification languages based on graph grammars lack a module concept which enables to structure large specifications. Such a concept is heavily needed to master the complexity of specifications of real-world applications. A first approach in this direction was proposed by Ehrig and Engels in [EE94, EE95]. The basic idea of this framework is to reuse structuring concepts which are known within or without the grammar field. These are the concept of distributed graph transformation systems, the concept of inheritance of specifications, and the import-export-interface concept (borrowed from the area of algebraic modular specifications). In [TS95], Taentzer and Schürr propose so-called DIEGO module systems which offer means for distributed programming with encapsulated graph objects. Their operational semantics definition follows the lines of distributed graph transformation as presented in [Tae95]. The concurrently interacting objects support information hiding and communicate by import-export interfaces.

GRACE is a new graph and rule centered language in an early stage of development involving researchers from Aachen, Berlin, Bremen and Leiden. It is intended as a specification and programming language for rule-based systems employing graph rewriting. The dream behind GRACE is to make the potentials of graph grammars and graph transformation systems more visible and applications of them more popular. Some ideas and principles of GRACE are outlined in Kreowski's paper [Kre95].

In [Kus95a], Kuske illustrates by an example the idea of structured programming with transformation units in GRACE. Moreover, two unary operations are presented, each of which constructs a normal form of a transformation unit. It is shown that none of these operations changes the interleaving semantics of the transformation unit it is applied to. Both operations underline the flexibility transformation units offer concerning the style of programming. Especially, they induce some necessary conditions under which the import of transformation units does not increase the expressive power.

In [KK95], Kreowski and Kuske introduce the notion of a *transformation unit* together with its interleaving semantics and study it as a means of constructing large graph transformation systems from small ones in a structured and systematic way. A transformation unit comprises a set of rules, descriptions of initial and terminal graphs, and a control condition. Moreover, it may import other transformation units. Its semantics is a binary relation between initial and terminal graphs which is given by interleaving sequences that obey the control condition. An interleaving sequence is a kind of structured derivation consisting of direct derivation steps interleaved with calls of imported transformation units. The introduced framework is independent of a particular graph transformation approach and hence may enhance the usefulness of graph transformation in many contexts.

Andries, Freund, and Stary [AFS94] propose a model of a graph grammar programming language as a unifying framework for formal software specification. The model can be customized for different application areas, but can also serve as an intermediate implementation platform for other high level graph grammar programming languages.

Engels and Schürr started to work on a concrete module concept for a graph grammar specification language [ES95a, ES95b]. This work is based on experiences from the software engineering field, where information hiding concepts like export and import interfaces are quite common. This has been combined with a new hierarchical graph data model resulting in encapsulated hierarchical graphs. Reusing the schema idea of the database world, hierarchical graphs are typed by schema graphs on a higher layer. Up to now only the structural aspects of such a new modular graph grammar specification approach has been investigated. The behavioural aspects, i.e., transformations of hierarchical graphs, are subject of current research.

## 4.4 Application to Language Definitions

Wagner and Gogolla [WG96] model the object based database language Troll Light by single-pushout graph transformation.

Andries and Engels investigate how graph grammars can be used to specify syntax as well as semantics of hybrid query languages for databases. In hybrid languages, the user can specify a query using an arbitrary mixture of graphical elements and text. The abstract syntax of the graphical part of the hybrid languages is formally specified by means of a PROGRES graph grammar. Semantics of the languages is defined by incorporating attribute evaluations into this graph grammar, which translate the graphical part into its textual equivalent. Results were presented in [AE94, AE95a, AE95b] and are elaborated upon in the forthcoming PhD thesis of M. Andries [And96].

The efforts of the past decade to provide designers and users with more powerful tools for both modeling the considered application domain and manipulating this model, has resulted in a variety of declarative data languages of usually high complexity. In view of the implementation and optimization of such languages, it is often beneficial to use a second language as an intermediate, machine independent implementation platform. In [AFS94, AFHS95], Andries, Freund, Haberstroh, and Starya introduce a formalism of graph rewriting systems as such an intermediate language. In [AFHS95], it is shown how the abstract implementation of a high level database language in this intermediate language supports the optimized implementation of specific database manipulation procedures.

Rekers and Schürr consider the definition of the syntax of visual languages and the parsing of pictures according to such syntax definitions. In [Rek94], it is advocated that pictures can be represented quite naturally by graphs and that graph grammars can be used to define the syntax of these graphs. Rekers and Schürr have developed a parsing algorithm for layered graph grammars, which can be used as such a parser of pictures. In the technical report [RS95b] the entire algorithm is explained and proved. Two versions of the paper were presented in different research communities: for the Graph Grammar community a quite formal version was presented at the Workshop on Graph Grammars and their Application to Computer Science (GraGra '94), see [RS94]. For the visual languages world a motivating version was presented at the 11th IEEE Symposium on Visual Languages (VL '95), see [RS95a].

## 4.5 Various Applications

Löwe, Korff, and Wagner [LKW91, LKW93] extend the single-pushout (SPO) approach to graph transformation to attributed graph transformation. The new theory is further developed in [Wag93]. In [Kor92] and [Kor94], conditional equations—functioning as application conditions—are added to the SPO approach. This kind of graph transformation is applied to actor systems in [Kor94].

In [TB94], Beyer and Taentzer use parallel graph transformation systems based on the double-pushout (DPO) approach to specify AGG, an algebraic graph grammar system. The AGG system, which implements the SPO-approach, is briefly described in [LB93] and [Bey93].

Graph-interpreted graph transformations were proposed as an adequate means for specifying concurrent object-oriented systems [Kor93a] and in more detail in [Kor95]. An alternative approach for specifying concurrent object-oriented systems using so-called object-oriented algebraic high-level nets is provided in [Rib94]. Following [HHT96], a combined framework for conditional rules, amalgamation concepts and transactions was proposed in [HMTW95]; a corresponding case study can be found in [CGH+95] where the graph editor AGG was specified. The aspect of the usefulness of amalgamation for the specification of the AGG-system is presented in [TB94]. In [Bet94a, Bet94b] graph transformations are used to support the software development process. In [Bet94a] this is done at an example of the development of a lift control system and in [Bet94b] the general methodology behind this process is explained. A filesystem and a medical information systems were specified using algebraic high-level nets in [REP93] and [PREC93] respectively. In [ELO95] it was shown how graph transformations can be considered as a tool for specifying dynamic abstract data types which were newly proposed as generalizations of the well-known concept of abstract data types. Shipping software was considered as a corresponding example in [EB94].

A case study where the graph editor AGG has been specified by single-pushout graph rewriting, as presented by Heckel, Müller, Taentzer, and Wagner [HMTW95], can be found in [CGHB+95]. This kind of graph transformation is also used in [Con95] where selected problems of knowledge representation are modelled. Both applications are valuated according to the possibilities and limitations of graph transformation.

In [Bet94b], Betschko uses graph transformation to support the software development process.

In [CGL96] graph transformations have been applied to the specification, construction and correctness of information systems.

Goal of the *GRIDS* project, as described by Zamperoni in [Zam95], is to provide a formally based, multi-dimensional software engineering model—and tool—that integrates "partial" models of software processes, system architectures, and views onto the system into one consistent project framework, in order to enhance real-life, large-scale software development. The static part of the so-called *Three-Dimensional Model of Software Engineering* (*3DM*) captures and structures partial models, integrated project frameworks, and other relevant project information. The dynamic part of the *3DM* provides the necessary actions to generate, manipulate and maintain the entities of the static part. The programmed graph rewriting system PROGRES offers a powerful means to formally specify the conceptual model of *GRIDS* as well as a set of tools to support specification and execution of the formal specifications of *GRIDS*. At the same time, specific "partial" models, to fit the special development situation at TNO, are developed and described with the *3DM* by Zamperoni, Gerritsen, and Bril [ZG94, ZGB95].

In an attempt to combine ideas from biology: evolution, natural growth and artificial neural networks, into one system to find good artificial neural network topology for a given task, a graph grammar formalism called G2L-systems [BKHSK93a, BKHSK93b, Boe94] was developed by Boers et al. This formalism is based on L-systems. Sets of production rules for this graph grammar formalism act as a growth mechanism for a modular artificial neural network topology. An evolutionary algorithm optimizes a population of sets of production rules to give the best possible network topology for a given task. New ideas of changing the topology of the artificial neural network during the training phase are investigated by Boers, Borst, and Sprinkhuizen-Kuiper [BBSK95]. This consists of using simple graph transformations, adding nodes and edges (or neurons and weights) at positions where a "computational deficiency" is detected in the artificial neural network. As it is currently unknown whether it is posible to encode these changes into the production rules that generated the initial topology, a Lamarckian approach seems not to be possible.

Applications of the algebraic graph grammar techniques to semantical networks are demonstrated and investigated in [EHK92]. The paper emphasizes the problem of the consistent definition of semantic networks and gives a rule-based type definition.

Jeltsch and Kreowski [JK91] have developed a grammatical inference algorithm which gets finite sets of sample graphs as inputs and constructs hyperedge replacement grammars as output. The inference mechanism is based on the decomposition of rules using edge-disjoint coverings of the right-hand sides of the rules.

Korff [Kor90, Kor91] applies graph transformations to production systems. Elaborated syntactical independency criteria, based on well-known properties, like parallel and sequential independence, lead to a considerable optimization of standard search algorithms for production systems.

In [TS91] robot configurations are modeled. Their movements and their dependencies are described with a two-level graph grammar, based on the double pushout approach. The first level describes elementary movements, which are composed on the second level. This allows the presentation of complex and structured objects, respectively robots. It is the abstract categorical description of the Berlin approach that faciliates this two-level method and leads to a very intuitive and well-structured discrete model for robotics.

Löwe [Löw90a] introduces an implementation method of algebraic specifications by graph transformation systems. The algebraic semantics given to a certain graph transformation system allows to combine algebraic and graphical implementation steps within a common framework. A corresctness notion is provided for so-called graphical specification presentations which possess an algebraic semantics. Small examples demonstrate that graphical implementations can considerably increase efficiency. A special problem of expression evaluation, namely garbage collection, can be advantagously tackled by applying single pushout derivations [LE91].

Graph grammar concepts are applied for the design of consistent states and manipulations modeling actions, transactions, and schedules for the simultaneous executions on a data base system in [EK80]. Especially the use of the Church-Rosser Theorems for graph grammars allows to handle synchronization problems for data base systemd in a multi-user environment. The concepts are apllied to a data base system for a libary.

In [ER80] a mathematical foundation for reasoning about the correctness and computational complexity of record handling algorithms, using the algebraic approach to graph grammars is proposed. A class od pattern matching and replacement rules for graphs is specified, such that the application 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, Church-Rosser Theorems can be used for proving that families of transformatins are well-behaved.

The application of graph grammars to computer processed control is investigated in [MW82]. The specification of a railroad system is given in terms of graph grammars, graph productions and graph derivation sequences. The use of graph grammars and graph grammar theory as a language tool for description and reasoning is investigated in this case study. Graph grammars support transparency in system design and lead to formal specifications that are theorecically and intuitively well-understood. The amalgamation concept is applied to synchronization of graph manipulations wthin this railroad system in [BFH87].

This extended approach of graph grammars with application conditions in [EH85] is applied to a small libary system in order to show the flexibility of this concept for the design of systems in computer science and related areas.

# Chapter 5

# References

The reference list is in alphabetical order according to citation names rather than according to author names. Please note that citation names such as [ACPS93], which are used to refer multiple authors, precede citation names such as [And96], which are used for the references with a single author.


[ACPS93] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *J. ACM*, 40:1134–1164, 1993.

[AE94] M. Andries and G. Engels. Syntax and Semantics of Hybrid Database Languages. In H. Ehrig and H.J. Schneider, editors, *Graph Transformations in Computer Science*, volume 776 of Lecture Notes in Computer Science, pages 19–36, 1994.

[AE95a] M. Andries and G. Engels. A Hybrid Query Language for the Extended Entity Relationship Model: revised version of TR 93-15. Technical Report 95-03, Leiden University, Dept. of Comp. Science, 1995. (Also available by anonymous ftp from `ftp.wi.leidenuniv.nl` as `/pub/CS/TechicalReports/1995/tr95-03.ps.gz`).

[AE95b] M. Andries and G. Engels. A Hybrid Query Language for the Extended Entity Relationship Model. *Journal of Visual Languages and Computing*, 1995. To appear.

[AFHS95] M. Andries, R. Freund, B. Haberstroh, and C. Stary. An Intermediate Language for Software Specification. In Waldemar W. Koczodaj, editor, *Proceedings of the 7th International Conference of Computing and Information (ICCI'95)*, Journal of Computing and Information, 822–846, July 1995.

[AFS94] M. Andries, R. Freund, and C. Stary. Software Specification based on Graph Grammars. *Preliminary Proc. International Workshop on Graph Grammars and Their Application to Computer Science '94*, 83–88, 1994.

[And96] M. Andries. *Graph Grammars for Visual Database Languages.* PhD thesis, Leiden University, The Netherlands, February 1996.

[BBSK95] E.J.W. Boers, M.V. Borst, and I.G. Sprinkhuizen-Kuiper. Evolving artificial neural networks using the "Baldwin effect". In D.W. Pearson, N.C. Steele, and R.F. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms, Proceedings of the International Conference in Alès, France*, 333–336. Springer-Verlag, 1995. Also as Technical Report 95-14, Leiden Unversity, Dept. of Computer Science, The Netherlands.

[BC87] M. Bauderon and B. Courcelle. Graph expressions and graph rewritings. *Math. Syst. Theory*, 20:83–127, 1987.

[BE95] H.L. Bodlaender and J. Engelfriet. Domino treewidth. In *Proc. WG'94*, Lecture Notes in Computer Science 903, pp.1–13, 1995. Also Tech. Report UU-CS-1994-11, Utrecht University, 1994.

[BEHL87] P. Böhm, H. Ehrig, U. Hummert, and M. Löwe. Towards distributed graph grammars. In H. Ehrig, M. Nagl, G. Rozenberg, and A. Rosenfeld, editors, *3rd Int. Workshop on Graph Grammars and their Application to Computer Science, LNCS 291*, Berlin, 1987. Springer Verlag.

[BFH87] P. Böhm, H.-R. Fonio, and A. Habel. Amalgamation of graph transformations: a synchronization mechanism. *Journal of Computer and System Science*, 34:377–408, 1987.

[BEL92] M. Beyer, H. Ehrig, and M. Löwe (eds.). Computing by graph transformation (COMPUGRAPH): Survey, results, and applications. Booklet for the presentation of ESPRIT BRWG No. 3299 (COMPUGRAPH) at CeBIT'92 in Hannover, 1992.

[BKHSK93a] E.J.W. Boers, H. Kuiper, B.L.M. Happel, and I.G. Sprinkhuizen-Kuyper. Biological metaphors in designing modular artificial neural networks. In S. Gielen and B. Kappen, editors, *Proceedings of the International Conference on Artificial Neural Networks*. Springer-Verlag, 1993.

[BKHSK93b] E.J.W. Boers, H. Kuiper, B.L.M. Happel, and I.G. Sprinkhuizen-Kuyper. Designing modular artificial neural networks. In H.A. Wijshoff, editor, *Proceedings of Computing Science in The Netherlands CSN'93*, pages 87–96, 1993. Also as Technical Report TR 93-24, available by anonymous ftp from `ftp.wi.leidenuniv.nl` as `/pub/CS/TechnicalReports/1993/tr93-24.ps.gz`.

[BM96] A. Bottreau and Y. Metivier. The Kronecker product and local computations in graphs. In *CAAP'95*. Lecture Notes in Comput. Sci., 1996.

[Bau91] M. Bauderon. Infinite hypergraphs I: Basic properties. *Theor. Comput. Sci.*, 82:177–214, 1991.

[Bau92] M. Bauderon. Infinite hypergraphs II: Systems of recursive equations. *Theor. Comput. Sci.*, 103:165–190, 1992.

[Bau94] M. Bauderon. A categorical approach to vertex replacement: the generation of infinite graphs. In *5th International Conference on Graph Grammars, Williamsburg*, 1994. To appear in LNCS.

[Bau95a] M. Bauderon. Parallel rewriting of graphs through the pullback approach. In *Proc. of SEGRAGRA'95 "Graph Rewriting and Computation", Electronic Notes of TCS, volume 2*. Elsevier Science, 1995. http://www.elsevier.nl/locate/entcs .

[Bau95b] M. Bauderon. A uniform approach to graph rewriting: the pullback approach. In *WG'95*, 1995. LNCS 1017, 101–115.

[Bet92] R. Betschko. Parallele Graphgrammatiken mit Synchronisation. Studienarbeit, Technical University of Berlin, Dep. of Comp. Sci., 1992.

[Bet94a] R. Betschko. Specifying communication based systems with algebraic attributed graph transformation. In G. Hommel, editor, *Contributions to the Quality of Communication-Based Systems — Berichte zum Kolloquium des Graduiertenkollegs*. Technical University of Berlin, 1994. technical report 94-27.

[Bet94b] R. Betschko. Supporting the software development process with algebraic graph transformations. *Preliminary Proc. International Workshop on Graph Grammars and Their Application to Computer Science '94*, 1994.

[Bey93] M. Beyer. *AGG An Algebraic Graph System, User Manual*. Technical University of Berlin, Department of Computer Science, 1993.

[Bil91] M. Billaud. ANIGRAF: An interactive system with priorities. In *Proc. of Rewriting techniques and applications (RTA)*, LNCS 488, pages 437–438, 1991.

[Boe94] E.J.W. Boers. Using L-systems as Graph Grammars: G2L-systems. *Preliminary Proc. International Workshop on Graph Grammars and Their Application to Computer Science '94*, 214–219, 1994.

[Bra83] F.-J. Brandenburg. The computational complexity of certain graph grammars. In *Proc. 6th GI Conference on Theoretical Computer Science*, Lecture Notes in Computer Science 145, 1983, pp.91-99

[Bra88] F.-J. Brandenburg. On polynomial time graph grammars. In *Proc. STACS 88*, Lecture Notes in Computer Science 294, 1988, pp.227-236

[CD96] A. Corradini and F. Drewes. (Cyclic) Term Graph Rewriting is adequate for Rational Parallel Term Rewriting. draft, 1996.

[CGH+95] M. Conrad, M. Gajewsky, R. Holl-Biniasz, M. Rudolf, J. Demuth, S. Weber, R. Heckel, J. Müller, G. Taentzer and A. Wagner. Graphische Spezifikation ausgewählter Teile von AGG – einem algebraischen Graphgrammatiksystem. Technical Report, 95-7, TU Berlin, 1995.

[CE95] B. Courcelle and J. Engelfriet. A logical characterization of the sets of hypergraphs defined by hyperedge replacement grammars. *Math. Syst. Theory*, 28:515–552, 1995.

[CEL+94a] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. Abstract Graph Derivations in the Double-Pushout Approach. In H.-J. Schneider and H. Ehrig, editors, *Proceedings of the Dagstuhl Seminar 9301 on Graph Transformations in Computer Science*, volume 776 of *Lecture Notes in Computer Science*, pages 86–103. Springer Verlag, 1994.

[CEL+94b] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. An event structure semantics for safe graph grammars. In E.-R. Olderog, editor, *Programming Concepts, Methods and Calculi*, IFIP Transactions A-56, pages 423–444. North-Holland, 1994.

[CEL+94c] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. Note on standard representation of graphs and graph derivations. In H.-J. Schneider and H. Ehrig, editors, *Proceedings of the Dagstuhl Seminar 9301 on Graph Transformations in Computer Science*, volume 776 of *Lecture Notes in Computer Science*, pages 104–118. Springer Verlag, 1994.

[CEL+96a] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and J. Padberg. The category of Typed Graph Grammars and its adjunctions with categories of derivations. In *Proceedings of the Fifth International Workshop on Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer Science. Springer Verlag, 1996. To appear.

[CEL+96b] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. An Event Structure Semantics for Graph Grammars with Parallel Productions. In *Proceedings of the Fifth International Workshop on Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer Science. Springer Verlag, 1996. To appear.

[CER79] V. Claus, H. Ehrig, and G. Rozenberg(eds.). Graph grammars and their application to computer science and biology. In *LNCS 79*. Springer Verlag, 1979.

[CER93] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *J. Comput. Syst. Sci.*, 46:218–270, 1993.

[CGHB+95] M. Conrad, M. Gajewsky, R. Holl-Biniasz, M. Rudolf, J. Demuth, S. Weber, R. Heckel, J. Müller, G. Taentzer, and A. Wagner. Graphische Spezifikation ausgewählter Teile von AGG — einem algebraischen Graphgrammatiksystem. Technical Report 95-7, TU-Berlin, 1995.

[CGL96] I. Claßen, M. Gogolla, and M. Löwe. Dynamics in information systems: Specification, construction, and correctness. Technical Report 96–01, Technische Universität Berlin, 1996.

[CH95] A. Corradini and R. Heckel. A compositional approach to structuring and refinement of typed graph grammars. In *Proc. of SEGRAGRA'95 "Graph Rewriting and Computation", Electronic Notes of TCS, volume 2.* Elsevier Science, 1995. http://www.elsevier.nl/locate/entcs .

[CM83] I. Castellani and U. Montanari. Graph-Grammars for Distributed Systems. In H. Ehrig, M. Nagl, and G. Rozenberg, editors, *Proceedings of the 2nd International Workshop on Graph-Grammars and Their Application to Computer Science*, volume 153 of *Lecture Notes in Computer Science*, pages 20–38. Springer Verlag, 1983.

[CM91] A. Corradini and U. Montanari. An Algebra of Graphs and Graph Rewriting. In *Proceedings of the $4^{th}$ Summer Conference on Category Theory and Computer Science (CTCS '91)*, volume 530 of *Lecture Notes in Computer Science*, pages 236–260. Springer Verlag, 1991.

[CM93] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theor. Comput. Sci.*, 109:49–82, 1993

[CM94] B. Courcelle and Y. Metivier. Coverings and minors: application to local computations in graphs. *European J. of Combinatorics*, 15:127–138, 1994.

[CM95] A. Corradini and U. Montanari. Specification of concurrent systems: from petri nets to graph grammars. In *Quality of Communication-Based Systems*, pages 35–52. Kluwer Academic Publishers, 1995.

[CMR94] A. Corradini, U. Montanari, and F. Rossi. An Abstract Machine for Concurrent Modular Systems: CHARM. *Theoret. Comput. Sci.*, 122:165–200, 1994.

[CMR96] A. Corradini, U. Montanari, and F. Rossi. Graph processes. *Fundamenta Informaticae*, 1996. to appear.

[CMR$^+$91] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, and M. Löwe. Logic Programming and Graph Grammars. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proceedings of the 4th International Workshop on Graph-Grammars and Their Application to Computer Science*, volume 532 of *Lecture Notes in Computer Science*, pages 221–237. Springer Verlag, 1991.

[CMR$^+$96] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. Algebraic approaches to graph transformation I: Basic concepts and double pushout approach. Chapter for the *Handbook of Graph Transformations*, Volume I: Foundations (G.Rozenberg, ed.), World Scientific, to appear

[CP95] B. Courcelle and A. Paries. Mineurs d'arbres avec racines. *RAIRO Inform. Théor. et Appl.*, 29:401–422, 1995.

[CR93] A. Corradini and F. Rossi. Hyperedge Replacement Jungle Rewriting for Term Rewriting Systems and Logic Programming. *Theoret. Comput. Sci.*, 109:7–48, 1993.

[CR95] A. Corradini and F. Rossi. Asymmetric Conflict in Graph Grammars. In F. Rosselló and G. Valiente, editors, *Proceedings Colloquium on Graph Transformation and its Application in Computer Science*, 1995. Technical Report UIB-DMI-B-19, University of the Balearic Islands.

[CR96] A. Corradini and F. Rossi. Synchronized Composition of Graph Grammar Productions. In *Proceedings of the Fifth International Workshop on Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer Science. Springer Verlag, 1996. To appear.

[CRPP91] A. Corradini, F. Rossi, and F. Parisi-Presicce. Logic Programming as Hypergraph Rewriting. In *Proc. CAAP '91*, volume 493 of *Lecture Notes in Computer Science*, pages 275–295. Springer Verlag, 1991.

[CS95] B. Courcelle and G. Sénizergues. The obstructions of a minor-closed set of graphs defined by a context-free grammar. Submitted to Discrete Maths.

[CW94] A. Corradini and D. Wolz. Jungle Rewriting: an abstract description of a lazy narrowing machine. In H.-J. Schneider and H. Ehrig, editors, *Proceedings of the Dagstuhl Seminar 9301 on Graph Transformations in Computer Science*, volume 776 of *Lecture Notes in Computer Science*, pages 119–137. Springer Verlag, 1994.

[Con95] M. Conrad. Anwendung algebraischer Graphtransformationen auf ausgewählte Probleme der Wissensrepräsentation und -verarbeitung. Technische Universität Berlin, 1995. Technical Report 95-11.

[Cor93] A. Corradini. Term Rewriting in $CT_\Sigma$. In *Proceedings CAAP '93*, volume 668 of *Lecture Notes in Computer Science*, pages 468–484. Springer Verlag, 1993.

[Cor95] A. Corradini. Concurrent computing: From petri nets to graph grammars. In A. Corradini and U. Montanari, editors, *Proceedings SEGRAGRA'95 Workshop on Graph Rewriting and Computation*, volume 2 of *Electronic Notes in Theoretical Computer Science*. Elsevier Sciences, 1995. tt http://www.elsevier.nl/locate/entcs/volume2.html.

[Cou87] B. Courcelle. An axiomatic definition of context-free rewriting and its application to NLC graph grammars. *Theor. Comput. Sci.*, 55:141–181, 1987.

[Cou89] B. Courcelle. The monadic second-order logic of graphs II: Infinite graphs of bounded width. *Math. Syst. Theory*, 21:187–221, 1989.

[Cou90a] B. Courcelle. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Vol. B*, J. Van Leeuwen ed., Elsevier, 1990, pp. 193-242.

[Cou90b] B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Inform. Comput.*, 85:12–75, 1990.

[Cou90c] B. Courcelle. The monadic second-order logic of graphs IV: Definability properties of equational graphs. *Annals Pure Applied Logic*, 49:193–255, 1990.

[Cou91] B. Courcelle. The monadic second-order logic of graphs V : On closing the gap between definability and recognizability. *Theor. Comput. Sci.*, 80:153–202, 1991.

[Cou92a] B. Courcelle. Graph grammars, monadic second-order logic and the theory of graph minors. In *Proc. AMS Conf. on Graph Minors*, Seattle, USA, 1992. AMS. Also: Bulletin of the EATCS 46, pp.193–226

[Cou92b] B. Courcelle. The monadic second-order logic of graphs III: Tree-decompositions, minors and complexity issues. *RAIRO Inform. Theor. et Applications*, 26:257–286, 1992.

[Cou92c] B. Courcelle. The monadic second-order logic of graphs VII : Graphs as relational structures. *Theor. Comput. Sci.*, 101:3–33, 1992.

[Cou94a] B. Courcelle. Monadic second-order logic and linear orderings of finite structures. In *Proc. 8th Workshop on Computer Science Logic.* LNCS 933, pp.163–176, 1995.

[Cou94b] B. Courcelle. The definition in ms logic of linear orderings of graphs and of modular decompositions of graphs. In *Proc. 5th International Graph Grammar Workshop.* LNCS, 1994. To appear.

[Cou94c] B. Courcelle. Monadic-second order definable graph transductions: a survey. *Theor. Comput. Sci.*, 126:53–75, 1994.

[Cou94d] B. Courcelle. The monadic second-order logic of graphs VI: On several representations of graphs by relational structures. *Discr. Appl. Math.*, 54:117–149, 1994.

[Cou95a] B. Courcelle. Basic notions of universal algebra for language theory and graph grammars. *Theor. Comput. Sci.*, 1995, to appear.

[Cou95b] B. Courcelle. The monadic second-order logic of graphs VIII: orientations. *Annals of Pure and Applied Logic*, 72:103–143, 1995.

[Cou95c] B. Courcelle. Structural properties of context-free sets of graphs generated by vertex replacement. *Inform. Comput.*, 116:275–295, 1995.

[Cou96a] B. Courcelle. *Handbook of graph transformations*, volume I: Foundations (G. Rozenberg, ed.), chapter The expression of graph properties and graph transformations in monadic second-order logic. World Scientific, 1996. To appear.

[Cou96b] B. Courcelle. The monadic second-order logic of graphs X: linear orders. *Theor. Comput. Sci.*, 1996. To appear.

[DDK93] G. David, F. Drewes, and H.-J. Kreowski. Hyperedge replacement with rendezvous. In J.P. Jouannaud, editor, *Proc. CAAP 93*, Lecture Notes in Computer Science 668, 167–181, 1993.

[DER94a] A.H. Deutz, A. Ehrenfeucht, and G. Rozenberg. Hyperedge channels are abelian. *Theor. Comput. Sci.*, 127:387-393, 1994.

[DER94b] A.H. Deutz, A. Ehrenfeucht, and G. Rozenberg. Clans and regions in 2-structures. *Theor. Comput. Sci.*, 129:207-262, 1994.

[DHK96] F. Drewes, A. Habel, and H.-J. Kreowski. Hyperedge replacement graph grammars. In *Handbook of Graph Transformations. Vol. I: Foundations* (G.Rozenberg, ed.). World Scientific. To appear.

[DHKT93] F. Drewes, A. Habel, H.-J. Kreowski, and S. Taubenberger. A sketch of collage grammars. *Bulletin of the EATCS* 50, 1993.

[DHKT95] F. Drewes, A. Habel, H.-J. Kreowski, and S. Taubenberger. Generating self-affine fractals by collage grammars. *Theor. Comput. Sci.*, 145:159–187, 1995.

[DHT96] J. Dassow, A. Habel, and S. Taubenberger. Chain-code pictures and collages generated by hyperedge replacement. In *Proc. International Workshop on Graph Grammars and Their Application to Computer Science '94*, Lecture Notes in Computer Science. Springer-Verlag. To appear.

[DK91] F. Drewes and H.-J. Kreowski. A note on hyperedge replacement. In [EKR91], pp. 1-11, 1991.

[DK96] F. Drewes and H.-J. Kreowski. (Un)decidability of properties of pictures generated by collage grammars. *in special issue of Fundamenta Informaticae.* To appear.

[DM87] P. Degano and U. Montanari. A model of distributed systems based on graph rewriting. *Journal of the ACM*, 34:411–449, 1987.

[DS93] I. Durand and B. Salinier. Constructor equivalent term rewriting systems. *Inf. Proc. Letters*, 47:131–137, 1993.

[DS94] I. Durand and B. Salinier. Constructor equivalent term rewriting systems are strongly sequential: a direct proof. *Inf. Proc. Letters*, 52:137–145, 1994.

[Dre93a] F. Drewes. NP-completeness of $k$-connected hyperedge-replacement languages of order $k$. *Inf. Proc. Letters* 45, 89–94, 1993.

[Dre93b] F. Drewes. Recognising $k$–connected hypergraphs in cubic time. *Theor. Comput. Sci.*, 109:83–122, 1993.

[Dre93c] F. Drewes. Transducibility — symbolic computation by tree-transductions. Report 2/93, Univ. Bremen, 1993.

[Dre94] F. Drewes. A lower bound on the growth of functions computed by tree transductions. In S. Tison, editor, *Proc. CAAP 94*, Lecture Notes in Computer Science 787, 100–114. Springer, 1994. Revised version to appear in *Fundamenta Informaticae* 26.

[Dre95a] F. Drewes. The use of tree transducers to compute translations between graph algebras. In *Proc. International Workshop on Graph Grammars and Their Application to Computer Science '94*, Lecture Notes in Computer Science. Springer-Verlag. To appear. (Long version in Report 8/94, Univ. Bremen, 1994.)

[Dre95b] F. Drewes. Semirings and tree-to-graph-to-tree transductions. In A. Corradini, U. Montanari, editors, *Proc. Joint* COMPUGRAPH/SEMAGRAPH *Workshop on Graph Rewriting and Computation (SEGRAGRA '95)*, volume 2 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1995.

[Dre96a] F. Drewes. On the connectedness of pictures defined by iterated function systems. In *Proc. 2nd Intl. Conference on Developments in Language Theory*. World Scientific, to appear.

[Dre96b] F. Drewes. Language theoretic and algorithmic properties of $d$-dimensional collages and patterns in a grid. *J. Comput. Syst. Sci.*, to appear.

[Dre96c] F. Drewes. Computation by tree transductions. Doctoral dissertation, University of Bremen, 1996.

[Dur94] I. Durand. Bounded, strongly sequential and forward-branching term rewriting systems. *Journal of Symbolic Computation*, 18:319–352, 1994.

[EB94] H. Ehrig and R. Bardohl. Specification techniques using dynamic abstract data types and application to shipping software. In *Proc. of the International Workshop on Advanced Software Technology*, pages 70–85, 1994.

[EBHL88] H. Ehrig, P. Böhm, U. Hummert, and M. Löwe. Distributed parallelism of graph transformation. In *13th Int. Workshop on Graph Theoretic Concepts in Computer Science, LNCS 314*, pages 1–19, Berlin, 1988. Springer Verlag.

[EE94] H. Ehrig and G. Engels. Pragmatic and semantic aspects of a module concept for graph transformation systems. In *Proceedings of the Fifth International Workshop on Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer Science. Springer Verlag, 1996. To appear.

[EE95] H. Ehrig and G. Engels. Towards a module concept for graph transformation systems: The software perspective. In G. Valiente Feruglio and F. Rosello Llompart, editors, *Proc. Colloquium on Graph Transformation and its Application in Computer Science*. Technical Report B-19, Universitat de les Illes Balears, 1995.

[EER96] A. Ehrenfeucht, J. Engelfriet, and G. Rozenberg. Finite languages for the representation of finite graphs. *J. Comput. Syst. Sci.*, 52:170–184, 1996.

[EH85] H. Ehrig and A. Habel. Graph grammars with application conditions. In G. Rozenberg and A. Salomaa, editors, *The Book of L*, pages 87–100. Springer Verlag, 1985.

[EH91] J. Engelfriet and L. Heyker. The string generating power of context-free hypergraph grammars. *J. Comp. Syst. Sci.*, 43:328–360, 1991.

[EH92] J. Engelfriet and L. Heyker. Context-free hypergraph grammars have the same term-generating power as attribute grammars. *Acta Informatica*, 29:161–210, 1992.

[EHK92] H. Ehrig, A. Habel, and H.-J. Kreowski. Introduction to graph grammars with applications to semantical networks. *Computers and Mathematics with Applications*, 23(6-9):557–572, 1992.

[EHKP91a] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. From graph grammars to High Level Replacement Systems. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *4th Int. Workshop on Graph Grammars and their Application to Computer Science, LNCS 532*, pages 269–291. Springer Verlag, 1991. Lecture Notes in Computer Science 532.

[EHKP91b] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. Parallelism and concurrency in high-level replacement systems. *Math. Struct. in Comp. Science*, 1:361–404, 1991.

[EHK$^+$96] H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner, and A. Corradini. Algebraic approaches to graph transformation II: Single pushout approach and comparison with double pushout approach. Chapter for the *Handbook of Graph Transformations*, Volume I: Foundations (G.Rozenberg, ed.), World Scientific, to appear

[EHL94] J. Engelfriet, L.M. Heyker, and G. Leih. Context-free graph languages of bounded degree are generated by apex graph grammars. *Acta Informatica*, 31:341-378, 1994.

[EHPR94] A. Ehrenfeucht, H.J. Hoogeboom, P. ten Pas, and G. Rozenberg. An introduction to context-free text grammars. In: *Developments in Language Theory*, pages 357-369. World Scientific, 1994.

[EHPR96] J. Engelfriet, T. Harju, A. Proskurowski, and G. Rozenberg. Characterization and complexity of uniformly non-primitive labeled 2-structures. *Theor. Comput. Sci.*, 154:247–282, 1996.

[EHR86] H. Ehrig, A. Habel, and B. Rosen. Concurrent Transformation of Relational Structures. *Fundamenta Informatica*, IX:13–50, 1986.

[EHR94a] A. Ehrenfeucht, T. Harju, and G. Rozenberg. Quotients and plane trees of group labeled 2-structures. In *Proc. 5th Workshop on Graph Grammars and Their Application to Computer Science*, 1994

[EHR94b] A. Ehrenfeucht, T. Harju, and G. Rozenberg. Invariants of inversive 2-structures on groups of labels. Tech. Report 94-08, Leiden University, 1994.

[EHR96] A. Ehrenfeucht, T. Harju, and G. Rozenberg. 2-Structures, Chapter for the *Handbook of Graph Transformations*, Volume I: Foundations (G.Rozenberg, ed.), World Scientific, to appear

[EK76] H. Ehrig and H.-J. Kreowski. Categorical theory of graphical systems and graph grammars. In *Conf. Report Algebraic System Theory, Udine 1975, Lecture Notes Econ. Math. Syst. 131 (1976)*, pages 323–351. Springer Verlag, 1976.

[EK76a] H. Ehrig and H.-J. Kreowski. Parallel graph grammars. In A. Lindenmayer and G. Rozenberg, editors, *Automata, Languages, Development*, pages 425–447. Amsterdam: North Holland, 1976.

[EK76b] H. Ehrig and H.-J. Kreowski. Parallelism of manipulations in multidimensional informatica structures. In *LNCS 45*, pages 284 – 293. Springer Verlag, 1976. Proc. Conf. Math. Foundations of Computer Science.

[EK79] H. Ehrig and H.-J. Kreowski. Pushout-properties: an analysis of gluing constructions for graphs. *Mathematische Nachrichten*, 91:135–149, 1979.

[EKL91] H. Ehrig, M. Korff, and M. Löwe. Tutorial introduction to the algebraic approach of graph grammars based on double and single pushouts. In [EKR91], pages 24–37. Lecture Notes in Computer Science 532.

[EKMS$^+$81] H. Ehrig, H.-J. Kreowski, A. Maggiolo-Schettini, B.K. Rosen, and J. Winkowski. Transformations of structures: an algebraic approach. *Math. Syst. Theory*, 14:305–334, 1981.

[EKR91] H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors. *4th International Workshop on Graph Grammars and Their Application to Computer Science.* Springer Verlag, 1991. Lecture Notes in Computer Science 532.

[EKT94] H. Ehrig, H.-J. Kreowski, and G. Taentzer. Canonical derivations for high level replacement systems. In *Lecture Notes in Computer Science 776*, pages 153–169. Springer Verlag, 1994. Also Technical Report 6/92, University of Bremen.

[EK80] H. Ehrig and H.-J. Kreowski. Applications of graph grammar theory to consistency, synchronization, and scheduling in database systems. *Information Systems*, 5:225–238, 1980.

[EK93] H. Ehrig and M. Korff. Computing by Graph Transformations II, ESPRIT BRWG COMPUGRAPH. *EATCS Bulletin*, (50):105–111, 1993.

[EK95] H. Ehrig and M. Korff. Computing with algebraic graph transformations: Overview of recent results. In G. Valiente Feruglio and F. Rosello Llompart, editors, *Proc. Colloquium on Graph Transformation and its Application in Computer Science.* Technical Report B-19, Universitat de les Illes Balears, 1995.

[EL89] H. Ehrig and M. Löwe(eds.). Computing by graph transformation (Gra Gra) — objectives, state-of-the-art and research issues, references). Technical Report 89/14, Technical University of Berlin, Dep. of Comp. Sci., 1989.

[EL89b] J. Engelfriet and G. Leih. Linear graph grammars: power and complexity. *Inform. Comput.*, 81:88-121, 1989.

[EL91] H. Ehrig and M. Löwe(eds.). Computing by graph transformation–overall aims and new results. In Ehrig et al. [EKR91]. also in EATCS-Bulletin 41, 1990.

[EL92a] H. Ehrig and M. Löwe. Computing by graph transformation — final report. Technical Report 92/08, Technical University of Berlin, Dep. of Comp. Sci., 1992.

[EL92b] H. Ehrig and M. Löwe. Computing by graph transformation (COMPUGRAPH): a survey. Technical Report 92/14, Technical University of Berlin, Dep. of Comp. Sci., 1992.

[EL92c] H. Ehrig and M. Löwe. From parallel to distributed derivations of graphs in the single pushout approach. *Memorial Volume for R. Frank, Inf. Fachberichte 309, Berlin*, pages 47–65, 1992.

[EL93] H. Ehrig and M. Löwe. Parallel and distributed derivations in the single pushout approach. *Theoretical Computer Science*, 109:123 – 143, 1993. Also in Tech. Rep. 91/01, Technical University of Berlin.

[EL93a] H. Ehrig and M. Löwe. Categorical principles, techniques and results for high-level replacement systems in computer science. *Applied Categorical Structures*, 1(1):21–50, 1993.

[ELO95] H. Ehrig, M. Löwe, and F. Orejas. Dynamic abstract data types based on algebraic graph transformations. In *Proc. ADT-Workshop '94, Springer LNCS, 906*, pages 236–254, 1995.

[ELR88] J. Engelfriet, G. Leih, and G. Rozenberg. Apex graph grammars and attribute grammars. *Acta Informatica*, 25:537-571, 1988.

[ELW90] J. Engelfriet, G. Leih, and E. Welzl. Boundary graph grammars with dynamic edge relabeling. *J. Comp. Syst. Sci.*, 40:307-345, 1990.

[ENR83] H. Ehrig, M. Nagl, and G. Rozenberg, editors. *2nd Int. Workshop on Graph Grammars and their Application to Computer Science, LNCS 153.* Springer, 1983.

[ENRR87] H. Ehrig, M. Nagl, G. Rozenberg, and A. Rosenfeld, editors. *3rd Int. Workshop on Graph Grammars and their Application to Computer Science, LNCS 291.* Springer Verlag, 1987.

[EO95] J. Engelfriet and V. van Oostrom. Regular description of context-free graph languages. Tech. Report 95-34, Leiden University, 1995.

[EP91a] H. Ehrig and F. Parisi-Presicce. Algebraic specification grammars: adjunction between module specifications and graph grammars. *Lecture Notes in Computer Science*, 532:292–310, 1991.

[EP91b] H. Ehrig and F. Parisi-Presicce. Nonequivalence of categories for equational algebraic specifications in view of high-level replacement systems. Technical Report 91/16, Technical University of Berlin, Dep. of Comp. Sci., 1991. Short version in Proc. 3rd Conf. on Algebraic and Logic Programming, Pisa, 1992.

[EP91c] H. Ehrig and F. Parisi-Presicce. A match operation for rule-based modular system design. In *Proc. ADT-Workshop, Wusterhausen, LNCS 534*, pages 74–97. Springer Verlag, 1991.

[EP93] H. Ehrig and F. Parisi-Presicce. Interaction between algebraic specification grammars and modular systems. In *Proc. AMAST 93*, 1993.

[EPR94] H. Ehrig, J. Padberg, and L. Ribeiro. Algebraic high-level nets: Petri nets revisited. In *Recent Trends in Data Type Specification*, pages 188–206, Caldes de Malavella, Spain, 1994. Springer Verlag. Lecture Notes in Computer Science 785.

[EPS73] H. Ehrig, M. Pfender, and H. J. Schneider. Graph grammars: an algebraic approach. In *14th Annual IEEE Symposium on Switching and Automata Theory*, pages 167–180, 1973.

[ER76] H. Ehrig and G. Rozenberg. Some definitional suggestions for parallel graph grammars. In A. Lindenmayer and G. Rozenberg, editors, *Automata, Laanguages Development*, pages 443 – 468. North-Holland Publishing Compasny, Yorktown Heights, 1976.

[ER78] H. Ehrig and B.K. Rosen. Concurrency of manipulations in multi-dimensional information structures. In *LNCS 64*, pages 65–176. Springer Verlag, 1978.

[ER80] H. Ehrig and B. K. Rosen. Parallelism and concurrency of graph manipulations. *Theoretical Computer Science*, 11:247–275, 1980.

[ER80] H. Ehrig and B.K. Rosen. The mathematics of record handling. In *SIAM J. COMPUT.*, pages 441–469. 1980.

[ER90a] A. Ehrenfeucht and G. Rozenberg. A Theory of 2-structures, Part I and II. *Theor. Comput. Sci.*, 70:277-342, 1990.

[ER90b] A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures, Part II: State spaces of concurrent systems. *Acta Informatica*, 27:343-368, 1990.

[ER90c] J. Engelfriet and G. Rozenberg. A comparison of boundary graph grammars and context-free hypergraph grammars. *Inform. Comput.*, 84:163-206, 1990.

[ER91] J. Engelfriet and G. Rozenberg. Graph grammars based on node rewriting: an introduction to NLC graph grammars. In [EKR91], pp. 12-23, 1991.

[ER94a] A. Ehrenfeucht and G. Rozenberg. Dynamic labeled 2-structures with variable domains. In: *Results and Trends in Theoretical Computer Science*, Lecture Notes in Computer Science 812, pages 97-123. 1994.

[ER94b] A. Ehrenfeucht and G. Rozenberg. Square systems. *Fundamenta Informaticae*, 20:75-111, 1994.

[ER95] A. Ehrenfeucht and G. Rozenberg. Dynamic labeled 2-structures, *Math. Struct. in Comp. Sci.*, 4:433-455, 1995.

[ER96] J. Engelfriet and G. Rozenberg. Node replacement graph grammars. Chapter for the *Handbook of Graph Transformations*, Volume I: Foundations (G.Rozenberg, ed.), World Scientific, to appear

[ES95a] G. Engels and A. Schürr. Encapsulated Hierarchical Graphs, Graph Types, and Meta Types. Technical Report 95-21, Leiden University, Dept. of Comp. Science, July 1995.

[ES95b] G. Engels and A. Schürr. Encapsulated Hierarchical Graphs, Graph Types, and Meta Types. In Andrea Corradini and Ugo Montanari, editors, *Proc. Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation.* Electronic Notes in Theoretical Computer Science. Elsevier, 1995. http://www.elsevier.nl/locate/entcs .

[ET92] H. Ehrig and G. Taentzer. From parallel graph grammars to parallel high- level replacement systems. In *Lindenmayer Systems*, pages 283–303. Springer Verlag, 1992.

[ET95] H. Ehrig and G. Taentzer. COMPUGRAPH II: a survey of research goals and main results. to appear in EATCS Bulletin, 1995.

[EV94] J. Engelfriet and H. Vogler. The translation power of top-down tree-to-graph transducers. *J. Comp. Syst. Sci.*, 49:258–305, 1994.

[EV95] J. Engelfriet and J.J. Vereijken. Context-free graph grammars and concatenation of graphs, Tech. Report 95-27, Leiden University, 1995

[EV96a] J. Engelfriet and J.J. Vereijken. Concatenation of graphs. In *Proc. Graph Grammars and Their Application to Computer Science*, Lecture Notes in Computer Science, Springer-Verlag, to appear

[EV96b] J. Engelfriet and H. Vogler. Bottom-up and top-down tree-to-graph transducers, in preparation, 1996

[Ehr77] H. Ehrig. Embedding theorems in the algebraic theory of graph grammars. In *LNCS 56*, pages 245–255. Springer Verlag, 1977.

[Ehr79] H. Ehrig. Introduction to the algebraic theory of graph grammars. In V. Claus, H. Ehrig, and G. Rozenberg, editors, *1st Graph Grammar Workshop, Lecture Notes in Computer Science 73*, pages 1–69. Springer Verlag, 1979.

[Ehr87] H. Ehrig. Tutorial introduction to the algebraic approach of graph grammars. In *3rd Int. Workshop on Graph Grammars and their Application to Computer Science*, LNCS 291, pages 3–14. Springer Verlag, 1987.

[Ehr95] H. Ehrig. Introduction to COMPUGRAPH. *Proc. of SEGRAGRA'95 "Graph Rewriting and Computation", Electronic Notes of TCS*, 2, 1995. http://www.elsevier.nl/locate/entcs .

[Eng89] J. Engelfriet. Context-free NCE graph grammars. In *Proc. FCT '89*, Lecture Notes in Computer Science 380, Springer-Verlag, Berlin, 1989, pp.148-161

[Eng91] J. Engelfriet. A characterization of context-free NCE graph languages by monadic second-order logic on trees. In [EKR91], pp. 311-327, 1991.

[Eng94] J. Engelfriet. Graph grammars and tree transducers. In *Proc. CAAP'94*, Lecture Notes in Computer Science 787, pp.15–36, 1994.

[HCEL95] R. Heckel, A. Corradini, H. Ehrig, and M. Löwe. Horizontal and vertical structuring of typed graph transformation systems. Submitted to special issue of MSCS, 1995.

[HHT96] A. Habel, R. Heckel, and G. Taentzer. Graph grammars with negative application conditions. Accepted for special issue of Fundamenta Informaticae, 1996.

[HK91] A. Habel and H.-J. Kreowski. Collage grammars. In *Proc. Graph Grammars and Their Application to Computer Science*, Lecture Notes in Computer Science 532, 411–429. Springer-Verlag, 1991.

[HKL93] A. Habel, H.-J. Kreowski, and C. Lautemann. A comparison of compatible, finite, and inductive graph properties. *Theor. Comput. Sci.* 110:145–168, 1993.

[HKP91] Annegret Habel, Hans-Jörg Kreowski, and Detlef Plump. Jungle evaluation. *Fundamenta Informaticae*, 15(1):37–60, 1991.

[HKT93] A. Habel, H.-J. Kreowski, and S. Taubenberger. Collages and patterns generated by hyperedge replacement. *Languages of Design* 1, 125–145, 1993.

[HKV91] A. Habel, H.-J. Kreowski, and W. Vogler. Decidable Boundedness Problems for Sets of Graphs Generated by Hyperedge-Replacement. *Theor. Comput. Sci.*, 89:33–62, 1991.

[HMTW95] R. Heckel, J. Müller, G. Taentzer, and A. Wagner. Attributed graph transformations with controlled application of rules. In G. Valiente Feruglio and F. Rosello Llompart, editors, *Proc. Colloquium on Graph Transformation and its Application in Computer Science.* Technical Report B-19, Universitat de les Illes Balears, 1995.

[HR94] T. Harju and G. Rozenberg. Decompositions of Infinite Labeled 2-Structures. In: *Results and Trends in Theoretical Computer Science*, Lecture Notes in Computer Science 812, pages 145-158, 1994.

[HP88] B. Hoffmann and D. Plump. Jungle evaluation for efficient term rewriting. In *Proc. Algebraic and Logic Programming.* Mathematical Research 49, pages 191–203, Berlin, 1988. Akademie-Verlag. Also in Springer Lecture Notes in Computer Science 343, 191–203, 1989.

[HP91] B. Hoffmann and D. Plump. Implementing term rewriting by jungle evaluation. *RAIRO Theoretical Informatics and Applications*, 25(5):445–472, 1991.

[HP95] A. Habel and D. Plump. Unification, rewriting, and narrowing on term graphs. In Andrea Corradini and Ugo Montanari, editors, *Proc. Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation.* Electronic Notes in Theoretical Computer Science. Elsevier, 1995. http://www.elsevier.nl/locate/entcs .

[HW95] R. Heckel and A. Wagner. Ensuring consistency of conditional graph grammars – a constructive approach. *Proc. of SEGRAGRA'95 Graph Rewriting and Computation, Electronic Notes of TCS*, 2, 1995. http://www.elsevier.nl/locate/entcs .

[Hab92a] A. Habel. *Hyperedge Replacement: Grammars and Languages*, volume 643 of Lecture Notes in Computer Science. Springer, 1992.

[Hab92b] A. Habel. Hypergraph grammars: Transformational and algorithmic aspects. *EIK – Journal on Information Processing and Cybernetics* 28(5), 241–277, 1992.

[Hec95a] R. Heckel. Algebraic graph transformations with application conditions. Master's thesis, TU-Berlin, 1995.

[Hec95b] R. Heckel. Embedding of conditional graph transformations. In G. Valiente Feruglio and F. Rosello Llompart, editors, *Proc. Colloquium on Graph Transformation and its Application in Computer Science*. Technical Report B-19, Universitat de les Illes Balears, 1995.

[Hof94] B. Hoffmann. Memoïzation for Term (Hyper) Graph Rewriting (extended abstract). Preliminary Proceedings of the 5th International Workshop on Graph Grammars and their Application to Computer Science, 1994.

[JK91] E. Jeltsch, H.-J. Kreowski: Grammatical Inference Based on Hyperedge Replacement. In Proc. *Graph Grammars and Their Application to Computer Science*. Lect. Not. Comp. Sci. 532, 461–474, 1991.

[JLR93] D. Janssens, M. Lens, and G. Rozenberg. Computation graphs for actor grammars. *Computer and System Sciences*, 46:60–90, 1993.

[JM95] D. Janssens and T. Mens. Abstract semantics for ESM systems. Technical Report 95–04, University of Antwerp - UIA, dept. of Math. and Comp. Sci., 1995. to appear in Fundamenta Informaticae.

[JR80] D. Janssens and G. Rozenberg. On the structure of node-label-controlled graph languages. *Information Sciences*, 20:191-216, 1980.

[JR89] D. Janssens and G. Rozenberg. Actor Grammars. *Mathematical Systems Theory*, 22:75–107, 1989.

[Jan93] D. Janssens: Equivalence of Computations in Actor Grammars, Theoret. Comput. Sci. 109 (1993) 145–180

[Jan94] D. Janssens. ESM systems and the composition of their computations. In *Graph Transformations in Computer Science*, volume 776 of *Lecture Notes in Computer Science*, pages 203–217. Springer Verlag, 1994.

[Jan95] D. Janssens. The decomposition of ESM computations. Technical Report 95–22, University of Antwerp - UIA, dept. of Math. and Comp. Sci., 1995.

[KEP95] P.W.M. Koopman, M.C.J.D. van Eekelen, and M.J. Plasmeijer. Operational Machine Specification in a Functional Programming Language. *Software Practice and Experience*, 25(5):463–499, May 1995.

[KK95] H.-J. Kreowski and S. Kuske. On the interleaving semantics of transformation units — A step into GRACE. In *Proc. International Workshop on Graph Grammars and Their Application to Computer Science '94*, Lecture Notes in Computer Science. Springer-Verlag. To appear.

[KKSdV90] J.R. Kennaway, J.W. Klop, M.R. Sleep, and F.-J. de Vries. Transfinite reductions in orthogonal Term Rewriting System. Technical Report Report CS-R9041, Centre for Mathematics and Computer Science, 1990.

[KR90] H.-J. Kreowski and G. Rozenberg. On structured graph grammars. Part I and II. *Information Science* 52, 185–210 and 221–246, 1990.

[KR94] M. Korff and L. Ribeiro. An attributed graph transformation approach to the behaviour of algebraic high-level nets. Extended abstract for the international Workshop on Graph Grammars'94, 1994.

[KR95a] M. Korff and L. Ribeiro. Concurrent derivations as single pushout graph grammar processes. In *Proc. of SEGRAGRA'95 "Graph Rewriting and Computation", Electronic Notes of TCS, volume 2*. Elsevier Science, 1995. http://www.elsevier.nl/locate/entcs .

[KR95b] M. Korff and L. Ribeiro. Formal relationship between graph grammars and Petri nets. In *5th Int. Workshop on Graph Grammars and their Application to Computer Science, Williamsburg '94, LNCS* , 1996. Accepted.

[KSEP93] P.W.M. Koopman, J.E.W. Smetsers, M.C.J.D. van Eekelen, and M.J. Plasmeijer. Graph Rewriting using the Annotated Functional Strategy. In M.R. Sleep, M.J. Plasmeijer, and M.C.J.D. van Eekelen, editors, *Term Graph Rewriting, Theory and Practice*. John Wiley, 1993.

[KZ94] P.W.M.Koopman and V. Zweije. Transformation of Functional Term Graph Rewrite Systems. Preliminary Proceedings of the 5th International Workshop on Graph Grammars and their Application to Computer Science, November 1994.

[Kau85] M. Kaul. *Syntaxanalyse von Graphen bei Präzedenz-Graph-Grammatiken.* Dissertation, Universität Osnabrück, 1985

[Kle96] R. Klempien-Hinrichs. Node replacement in hypergraphs: simulation of hyperedge replacement, and decidability of confluence. In *Proc. Graph Grammars and Their Application to Computer Science '94*, Lecture Notes in Computer Science. Springer-Verlag. To appear.

[Koo93] P.W.M. Koopman. Rewriting Term Graph Rewrite Systems. In H.A. Wijshoff, editor, *Proceedings of Computing Science in The Netherlands CSN'93*, 1993. ISBN 906196430X.

[Koo94] P.W.M. Koopman. Restricted data-types. In J. Glauert, editor, *Proceedings of the 6th international workshop on the implementation of functional languages*, September 1994. UEA, Norwich, UK.

[Kor90] M. Korff. Optimizations of production systems based on algebraic graph transformations. Technical report, Technical University of Berlin, Dep. of Comp. Sci., 1990.

[Kor91] M. Korff. Application of graph grammars to rule-based systems. In Lecture Notes in Computer Science 532, pages 505–519, 1991.

[Kor92] M. Korff. Algebraic transformations of equationally defined graph structures. Technical Report 92/32, Technical University of Berlin, 1992.

[Kor93] M. Korff. An algebraic graph grammar model for actor systems. Technical Report 93/21, Technical University of Berlin, 1993.

[Kor93a] M. Korff. Single pushout transformations of generalized graph structures. Technical Report RP 220, Federal University of Rio Grande do Sul, Porto Alegre, Brazil, 1993.

[Kor94] M. Korff. Single pushout transformations of equationally defined graph structures with applications to actor systems. In *Proc. Graph Grammar Workshop Dagstuhl 93*, pages 234–247. Springer Verlag, 1994. Lecture Notes in Computer Science 776.

[Kor94a] M. Korff. Graph-interpreted graph transformations for concurrent object-oriented systems. Extended abstract for the *5th International Workshop on Graph Grammars and their Application to Computer Science*, 1994.

[Kor94b] M. Korff. True concurrency semantics for single pushout graph transformaitons with applications to actor systems. In *Working papers of the International Workshop on Information Systems – Correctness and Reusability IS-CORE'94*, pages 244–258, 1994. Tech. Report IR-357, Free University, Amsterdam.

[Kor94c] M. Korff. Minimality of derived rules in single pushout graph rewriting. Technical Report 94/10, Technical University of Berlin, 1994.

[Kor95] M. Korff. True concurrency semantics for single pushout graph transformations with applications to actor systems. In R. J. Wieringa and R. B. Feenstra, editors, *Working papers of the International Workshop on InformaInformation Systems - Correctness and Reusability*, pages 33–50. World Scientific, 1995.

[Kor95] M. Korff. Graph-interpreted graph transformations for concurrent object-oriented systems. In *5th Int. Workshop on Graph Grammars and their Application to Computer Science, Williamsburg '94, LNCS* , 1996. Accepted.

[Kre77] H.-J. Kreowski. *Manipulation von Graphmanipulationen.* PhD thesis, Technische Universität Berlin, 1977.

[Kre77a] H.-J. Kreowski. Transformation of derivation sequences in gragra. In *LNCS 56*, pages 275–286. Springer Verlag, 1977.

[Kre81] H.-J. Kreowski. A comparison between Petri-nets and graph grammars. In *Lecture Notes in Computer Science 100*, pages 1–19. Springer Verlag, 1981.

[Kre92] Hans-Jörg Kreowski. Parallel hyperedge replacement. In Grzegorz Rozenberg and Arto Salomaa, editors, *Lindenmayer Systems*, pages 271–282. Springer-Verlag, 1992.

[Kre93a] H.-J. Kreowski. Five facets of hyperedge replacement beyond context-freeness. In *Proc. FCT 93*, Lecture Notes in Computer Science 710, pp.69–86, 1993.

[Kre93b] H.-J. Kreowski. Translations into the graph grammar machine. In Ronan Sleep, Rinus Plasmeijer, Marko van Eekelen, editors, *Term Graph Rewriting: Theory and Practice*, Chapter 13. John Wiley, 1993.

[Kre94] Hans-Jörg Kreowski. An axiomatic approach to canonical derivations. In *Proc. 13th IFIP World Computer Congress 1994*, pages 348–353, Elsevier Science B.V. (North Holland), 1994.

[Kre95] Hans-Jörg Kreowski. Graph grammars for software specification and programming: An eulogy in praise of GRACE. In G. Valiente Feruglio and F. Rosello Llompart, editors, *Proc. Colloquium on Graph Transformation and its Application in Computer Science.* Technical Report B-19, Universitat de les Illes Balears, 1995.

[Kus93] S. Kuske. A maximum path length pumping lemma for edge-replacement languages. In *Proc. FCT 93*, Lecture Notes in Computer Science 710, pp.342–335, 1993.

[Kus95a] S. Kuske. Semantic aspects of the graph and rule centered language GRACE. In G. Valiente Feruglio and F. Rosello Llompart, editors, *Proc. Colloquium on Graph Transformation and its Application in Computer Science.* Technical Report B-19, Universitat de les Illes Balears, 1995.

[Kus95b] S. Kuske. Implementing $\beta$-reduction by hypergraph rewriting. In *Proc. of SEGRAGRA'95 "Graph Rewriting and Computation", Electronic Notes of TCS, volume 2.* Elsevier Science, 1995. http://www.elsevier.nl/locate/entcs .

[LB93] M. Löwe and M. Beyer. AGG — an implementation of algebraic graph rewriting. In *Proc. Fifth Int. Conf. Rewriting Techniques and Applications, '93, LNCS 690*, pages 451–456. Springer Verlag, 1993.

[LD94] M. Löwe and J. Dingel. Parallelism in single-pushout graph rewriting. *Lecture Notes in Computer Science 776*, pages 234–247, 1994.

[LE91] M. Löwe and H. Ehrig. Algebraic approach to graph transformation based on single pushout derivations. In R.H. Möhring, editor, *Graph-Theoretic Concepts in Computer Science, WG '90*, pages 338–353. Springer LNCS 484, 1991.

[LKW91] M. Löwe, M. Korff, and A. Wagner. Single-pushout transformation of attributed graphs: a link between graph grammars and abstract data types. In *Proc. of the SEMAGRAPH Symposium 1991*, pages 359–379. Technical Report 91-25, University of Nijmegen, 1991.

[LKW93] M. Löwe, M. Korff, and A. Wagner. An algebraic framework for the transformation of attributed graphs. In M.R. Sleep, M.J. Plasmeijer, and M.C. van Eekelen, editors, *Term Graph Rewriting: Theory and Practice*, chapter 14, pages 185–199. John Wiley & Sons Ltd, 1993.

[LM92] I. Litovsky and Y. Metivier. Computing trees with graph rewriting systems with priorities. In M. Nivat and A. Podelski, editors, *Tree automata and languages*, pages 115–139. Elsevier Science Publishers B.V., 1992.

[LM93] I. Litovsky and Y. Metivier. Computing with graph rewriting systems with priorities. *Theor. Comput. Sci.*, 115:191–224, 1993.

[LM95] M. Löwe and J. Müller. Critical pair analysis in single-pushout graph rewriting. In G. Valiente Feruglio and F. Rosello Llompart, editors, *Proc. Colloquium on Graph Transformation and its Application in Computer Science.* Technical Report B-19, Universitat de les Illes Balears, 1995.

[LMS95a] I. Litovsky, Y. Metivier, and E. Sopena. Different local controls for graph relabelling systems. *Math. Syst. Theory*, 28:41–65, 1995.

[LMS95b] I. Litovsky, Y. Metivier, and E. Sopena. Checking global graph properties by means of local computations: the majority problem. *Electronic Notes in Theoretical Computer Science 1*, pages 51–58, 1995.

[LMZ95] I. Litovsky, Y. Metivier, and W. Zielonka. On the recognition of families of graphs with local computations. *Inform. Comput.*, 118:110–119, 1995.

[Lap94] D. Lapoire. Sur la largeur arborescente d'une graphe planaire et celle de son dual. Submitted, 1994.

[Lau88] C. Lautemann. Decomposition trees: structured graph representation and efficient algorithms. In *Proc. CAAP '88*, Lecture Notes in Computer Science 299, pages 28–39, Springer-Verlag, 1988.

[Lau90] C. Lautemann. The complexity of graph languages generated by hyperedge replacement. *Acta Informatica*, 27:399-421, 1990.

[Löw90] M. Löwe. Algebraic approach to graph transformation based on single pushout derivations with partial morphisms. Technical Report 90/5, Technical University of Berlin, Dep. of Comp. Sci., 1990.

[Löw90a] M. Löwe. Implementing algebraic specifications by graph transformations. *Journal for Information Processing and Cybernetics (EIK)*, 26(11/12):615–641, 1990.

[Löw90b] M. Löwe. *Extended Algebraic Graph Transformations*. PhD thesis, Technical University of Berlin, 1990.

[Löw93] M. Löwe. Algebraic approach to single-pushout graph transformation. *TCS*, 109:181–224, 1993.

[MM90] J. Meseguer and U. Montanari. Petri nets are monoids. *Info. and Co.*, 88:105–155, 1990.

[MR93] U. Montanari and F. Rossi. Graph rewriting for a partial ordering semantics of concurrent constraint programming. *Theoret. Comput. Sci.*, 109:225–256, 1993.

[MR94] U. Montanari and F. Rossi. Contextual occurrence nets and concurrent constraint programming. In H.-J. Schneider and H. Ehrig, editors, *Proceedings of the Dagstuhl Seminar 9301 on Graph Transformations in Computer Science*, volume 776 of *Lecture Notes in Computer Science*. Springer Verlag, 1994.

[MR95] U. Montanari and F. Rossi. Contextual nets. *Acta Informatica*, 32, 1995. Also as Technical Report TR 4-93, Department of Computer Science, University of Pisa, February 1993.

[MS94] Y. Metivier and N. Saheb. Probabilistic analysis of an election algorithm in a tree. In *CAAP'94*, volume 787, pages 234–246. Lecture Notes in Comput. Sci., 1994.

[MSNT88] A. Maggiolo-Schettini, M. Napoli, and G. Tortora. Web Structures: a Tool for Representing and Manipulating Programs. *IEEE Transactions on Software Engineering*, pages 1621–1639, 1988.

[MSP92] A. Maggiolo-Schettini and A. Peron. Semantics of statecharts based on graph rewriting. In *Proceedings of Advanced Research Workshop on Correct Hardware Design Methodologies*. Elsevier, 1992.

[MSP93] A. Maggiolo-Schettini and A. Peron. Semantics of full Statecharts based on graph rewriting. Technical Report 11-93/RR, UDMI Research Report, 1993.

[MSP94] A. Maggiolo-Schettini and A. Peron. Semantics of full Statecharts based on graph rewriting. In *Proc. Graph Grammar Workshop Dagstuhl 93*, pages 265–279. Springer, 1994. Lecture Notes in Computer Science 776.

[MSP96] A. Maggiolo-Schettini and A. Peron. A graph rewriting framework for Statecharts semantics. In *Proceedings of the Fifth International Workshop on Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer Science. Springer Verlag, 1996. To appear.

[MSW82] A. Maggiolo-Schettini and J. Winkowski. Processes of transforming structures. *J. Comput. Syst. Sci.*, 24:245–282, 1982.

[MSW83] A. Maggiolo-Schettini and J. Winkowski. Towards a Programming Language for Manipulating Relational Data Bases. In D. Bjorner, editor, *Formal Description of Programming Concepts - II*, pages 265–278. North-Holland, 1983.

[MSW91] A. Maggiolo-Schettini and J. Winkowski. Programmed derivations of relational structures. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proceedings of the 4th International Workshop on Graph-Grammars and Their Application to Computer Science*, volume 532 of *Lecture Notes in Computer Science*, pages 582–598. Springer Verlag, 1991.

[MSW92] A. Maggiolo-Schettini and J. Winkowski. A programming language for deriving hyper-graphs. In *Proceedings 17th Colloquium on Trees in Algebra and Programming (CAAP)*, volume 581 of *Lecture Notes in Computer Science*, pages 221–231. Springer Verlag, 1992.

[MSW94] A. Maggiolo-Schettini and J. Winkowski. A formalism for programmed rewriting of (hyper)graphs. Technical report, IPI-PAN Research Report, July 1994.

[MSW95] A. Maggiolo-Schettini and J. Winkowski. A kernel language for programmed rewriting of (hyper)graphs. *Acta Informatica*, 1995. In press.

[MSW96] A. Maggiolo-Schettini and J. Winkowski. Dynamic graphs. Technical report, IPI-PAN Research Report, 1996. To appear.

[MW82] B. Mahr and A. Wilharm. Graph grammars as a tool for description in computer processed control: A case study. In *Int. Workshop WG '82*, pages 165–176, München, Wien, 1982. Hanser Verlag.

[Mos96a] M. Mosbah. Probabilistic graph grammars. To appear in *special issue of Fundamenta Informaticae*, 1996.

[Mos96b] M. Mosbah. Probabilistic hyperedge replacement. *Theor. Comput. Sci.*, 159, 1996.

[Nag79] M. Nagl. *Graph-Grammatiken.* Vieweg, Braunschweig, 1979

[Nag80] M. Nagl. A tutorial and bibliographical survey of graph grammars, in *Graph Grammars* (V.Claus, H.Ehrig, G.Rozenberg, eds.). Lecture Notes in Computer Science 73, Springer-Verlag, Berlin, 1980, pp.70-126

[PER95] J. Padberg, H. Ehrig, and L. Ribeiro. Algebraic high-level net transformation systems. *Math. Struct. in Comp. Science*, 5:217–256, 1995.

[PH96] D. Plump and A. Habel. Graph unification and matching. In *Proc. International Workshop on Graph Grammars and Their Application to Computer Science '94*, Lecture Notes in Computer Science. Springer-Verlag. To appear.

[PPEM87] F. Parisi-Presicce, H. Ehrig, and U. Montanari. Graph rewriting with unification and composition. In *3rd Int. Workshop on Graph Grammars and their Application to Computer Science, LNCS 291*, Berlin, 1987. Springer Verlag.

[PREC93] J. Padberg, L. Ribeiro, H. Ehrig, and F. Cornelius. Formal requirement analysis using algebraic high-level nets and transformations: a KorSo case study improving parts of HDMS-A. Technical Report 93/34, Technical University of Berlin, 1993.

[PT95] J. Padberg and G. Taentzer. Embedding of derivations in high-level replacement systems. In G. Valiente Feruglio and F. Rosello Llompart, editors, *Proc. Colloquium on Graph Transformation and its Application in Computer Science.* Technical Report B-19, Universitat de les Illes Balears, 1995.

[Pad82] P. Padawitz. Graph grammars and operational semantics. *Theor. Comput. Sci.*, 19:37–58, 1982.

[Pad95] J. Padberg. Abstract datatype semantics for algebraic high-level nets using dynamic abstract datatypes. In G. Hommel, editor, *Proc. Int. Workshop on Quality of Communication Based Systems*, pages 1 – 18. Kluwer Academic Publishing, 1995.

[Pad96] J. Padberg. *Abstract Petri Nets: A Uniform Approach and Rule-Based Refinement.* PhD thesis, Technical University Berlin, 1996.

[Pel96] L. Pelecq. On automorphism groups of context-free graphs. To appear in *Theor. Comput. Sci.*, 1996.

[Plu91a] D. Plump. Implementing Term Rewriting by Graph Reduction: Termination of Combined Systems. In *Proc. 2nd Int. Workshop on Conditional and Typed Rewriting Systems.* Lect. Not. Comp. Sci. 516, 307–317, 1991.

[Plu91b] D. Plump. Graph-Reducible Term Rewriting Systems. In *Proc. Graph Grammars and Their Application to Computer Science.* Lect. Not. Comp. Sci. 532, 622–636, 1991.

[Plu93] D. Plump. Hypergraph rewriting: Critical pairs and undecidability of confluence. In Ronan Sleep, Rinus Plasmeijer, Marko van Eekelen, editors, *Term Graph Rewriting: Theory and Practice*, Chapter 15. John Wiley, 1993.

[Plu93a] D. Plump. Collapsed tree rewriting: Completeness, confluence, and modularity. In Proc. Conditional Term Rewriting Systems, *Lecture Notes in Computer Science* 656, 97–112, 1993.

[Plu93b] D. Plump. Evaluation of functional expressions by hypergraph rewriting. Dissertation, Univ. Bremen, 1993.

[Plu94] D. Plump. Critical pairs in term graph rewriting. In I. Prívara, B. Rovan, P. Ružička, editors, *Proc. Mathematical Foundations of Computer Science 1994*, Lecture Notes in Computer Science 841, pages 556–566, Springer-Verlag, 1994.

[Plu95] D. Plump. On termination of graph rewriting. In *Proc. 21st International Workshop on Graph-Theoretic Concepts in Computer Science (WG '95)*, Lecture Notes in Computer Science 1017, pages 88–100, Springer-Verlag, 1995.

[REP93] L. Ribeiro, H. Ehrig, and J. Padberg. Formal development of concurrent systems using algebraic high-level nets and transformations. In *Proc. VII Simpósio Brasileiro de Engenharia de Software*, pages 1–16, Rio de Janeiro, oct. 1993. Tech-report no. 93-13, TU Berlin.

[RP94] L. Ribeiro and J. Padberg. Algebraic high-level nets and transformations with initial markings. Technical Report 94/7, Technical University of Berlin, 1994.

[RW86] G. Rozenberg and E. Welzl. Boundary NLC graph grammars — basic definitions, normal forms, and complexity. *Inform. and Control* 69:136-167, 1986.

[RS94] J. Rekers and A. Schürr. A parsing algorithm for context-sensitive graph grammars (short version). In *Proc. International Workshop on Graph Grammars and Their Application to Computer Science '94*, Lecture Notes in Computer Science. Springer-Verlag. To appear. This paper is a shortened version of [RS95b].

[RS95a] J. Rekers and A. Schürr. A graph grammar approach to graphical parsing. In H.-J. Hoffmann, editor, *Proc. VL'95 11th International IEEE Symposium on Visual Languages*, 195–202, Darmstadt, Germany, September 1995. Available via ftp from `ftp.wi.leidenuniv.nl` as `/pub/CS/TechnicalReports/1995/tr95-15.ps.gz`.

[RS95b] J. Rekers and A. Schürr. A parsing algorithm for context-sensitive graph grammars (long version). Technical Report 95-05, Leiden University, Dept. of Computer Science, the Netherlands, 1995. Available via ftp from `ftp.wi.leidenuniv.nl` as `/pub/CS/TechnicalReports/1995/tr95-05.ps.gz`.

[Rao93] J.C. Raoult. Recursively defined tree-transductions. In *Proc. 5th RTA*, Lecture Notes in Computer Science 690, pp.343–357, 1993.

[Rek94] J. Rekers. On the use of Graph Grammars for defining the Syntax of Graphical Languages. In G. Valiente Feruglio and F. Rosello Llompart, editors, *Proc. Colloquium on Graph Transformation and its Application in Computer Science*. Technical Report B-19, Universitat de les Illes Balears, 1995. Also available by anonymous ftp from `ftp.wi.leidenuniv.nl`, */pub/CS/TechnicalReports/1994/tr94-11.ps.gz*.

[Rek95] J. Rekers. A definition of the graphical syntax of basic message sequence charts. Leiden University, The Netherlands, 1995.

[Rib94] L. Ribeiro. Object-oriented algebraic high-level nets. In *Working papers of the Int. Workshop on Information Systems: Correctness and Reusability, IS-CORE'94*, 1994. Tech. Report IR-357, Free University, Amsterdam.

[SE76] H.J. Schneider and H. Ehrig. Grammars on partial graphs. *Acta Informatica*, 6:297–316, 1976.

[SE94] H.J. Schneider and H. Ehrig(eds.): Graph Transformations in Computer Science. Lecture Notes in Computer Science 776, Springer, 1994

[SW95] K. Skodinis and E. Wanke. Emptiness problems of eNCE graph languages. *J. Comp. Syst. Sci.*, 51:472-485, 1995.

[SW96] K. Skodinis and E. Wanke. The bounded degree problem for eNCE graph grammars. *5th International Workshop on Graph-Grammars and their Application to Computer Science*, Williamsburg, Virginia, USA, November 1994.

[Sén93] G. Sénizergues. An effective version of Stallings' theorem in the case of context-free graphs. In *ICALP'93*, volume 700, pages 478–495. Lect. Notes. Comput. Sci., 1993.

[Sén96] G. Sénizergues. On the finite subgroups of a context-free group. In R. Gilman et al., editor, *Geometric and computational perspectives on infinite groups*, pages 1–11. American Mathematical Society, 1996. To appear.

[TB94] G. Taentzer and M. Beyer. Amalgamated graph transformation systems and their use for specifying AGG – an algebraic graph grammar system. Springer Verlag, 1994.

[TS91] G. Taentzer and H. Schween. Movements of objects in configuration spaces modelled by graph grammars. In Proc. *4th International Workshop on Graph Grammars and Their Application to Computer Science*, 1991.

[TS95] G. Taentzer and A. Schürr. DIEGO, another step towards a module concept for graph transformation systems. *Proc. of SEGRAGRA'95 "Graph Rewriting and Computation", Electronic Notes of TCS*, 2, 1995. http://www.elsevier.nl/locate/entcs .

[Tae96] G. Taentzer. Parallel high-level replacement systems. To appear in *Theor. Comput. Sci.*, 1996.

[Tae95] G. Taentzer. Hierarchically distributed graph transformation. In *Proc. of 5. Int. Workshop on Graph Grammars and Their Applications to Computer Science, LNCS, Springer, Berlin*, 1995. To appear.

[Tae96b] G. Taentzer. Towards synchronous and asynchronous graph transformations. accepted for special issue of Fundamenta Informaticae, 1996.

[Tae96c] G. Taentzer. *Parallel and Distributed Graph Transformation: Formal Description and Application to Communication-Based Systems.* PhD thesis, Technical University of Berlin, Dep. of Comp. Sci., 1996.

[Tau94] S. Taubenberger. Correct translations of generalized iterated function systems to collage grammars. Technical Report 7/94, Universität Bremen, Fachbereich Mathematik und Informatik, 1994.

[Vog91] W. Vogler. Recognizing edge replacement graph languages in cubic time. In [EKR91], 676-687

[Vog93] W. Vogler. On hyperedge replacement and BNLC graph grammars. *Discr. Appl. Math.*, 46:253-273, 1993.

[WG96] A. Wagner and M. Gogolla. Defining operational behavior of object specifications by attributed graph transformations. *in special issue of Fundamenta Informaticae.* To appear.

[WMS92] J. Winkowski and A. Maggiolo-Schettini. A programming language for concurrent rewriting of hypergraphs. Technical Report 719, IPI-PAN Research Report, 1992.

[Wag93] A. Wagner. Vergleich von High-Level-Replacement-Systemen basierend auf dem Double- bzw. Single-Pushout-Ansatz. Master's thesis, Technical University of Berlin, Dep. of Comp. Sci., 1993.

[Wag95] A. Wagner. On the expressive power of algebraic graph grammars with application conditions. In *Int. Joint Conf. on Theory and Practice of Software Development (TAPSOFT'95)*, LNCS 915. Springer Verlag, 1995.

[Wan91] E. Wanke. Algorithms for graph problems on BNLC structured graphs. *Inform. Comput.*, 94:93-122, 1991.

[Wan94] E. Wanke. The complexity of connectivity problems on context-free graph languages. *J. of Comp. Syst. Sci.*, 49:57-82, 1994.

[ZG94] A. Zamperoni and B. Gerritsen. Integrating the Developers' and the Managerial Perspective of an Incremental Development Life Cycle. In *Proc. of the 12th Annual Pacific Northwest Software Quality Conference, Portland, USA*, 227–242, October 1994. Also as Technical Report 94-22, Leiden University, Dept. of Comp. Science, The Netherlands (Available by *anonymous ftp* from `ftp.wi.leidenuniv.nl` as `/pub/CS/TechnicalReports/1994/tr94-22.ps.gz`).

[ZGB95] A. Zamperoni, B. Gerritsen, and B. Bril. Evolutionary Software Development: An Experience Report on Technical and Strategic Requirements. Technical Report 95-25, Leiden University, Dept. of Comp. Science, 1995. (Also available by *anonymous ftp* from `ftp.wi.leidenuniv.nl` as `/pub/CS/TechnicalReports/1995/tr95-25.ps.gz`).

[ZK93] V. Zweije and P.W.M. Koopman. Optimization of Functional Graph Rewrite Systems by Partial Evaluation. In Rinus Plasmeijer and Marko van Eekelen, editors, *Implementation of Functional Languages, Proceedings of the Fifth International Workshop on Implementation of Functional Languages*, September 1993.

[Zam95] A. Zamperoni. GRIDS — Graph-based, Integrated Development of Software: Integrating Different Perspectives of Software Engineering. Technical Report 95-24, Leiden University, Dept. of Comp. Science, 1995. (Also available by *anonymous ftp* from `ftp.wi.leidenuniv.nl` in `/pub/CS/TechnicalReports/1995/tr95-24.ps.gz`).