COMPUGRAPH II: A Survey of Research Goals and Main Results

Hartmut Ehrig and Gabriele Taentzer
Fachbereich Informatik, Technische Universität Berlin
D-10587 Berlin, Germany

Abstract

This is a survey of the main aims and results of the ESPRIT Basic Research Working Group COMPUTING BY GRAPH TRANSFORMATION II, 1992 - 1996, following up the first phase of COMPUGRAPH, 1989 - 1992. The research goals and main results are presented within the following three research areas: Foundations, Concurrency, and Graph Transformations for Specification and Programming.

1 Introduction

The research area of graph grammars or graph transformations is a relatively young discipline of computer science. Its origins date back to the early seventies. Nevertheless, 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 "programming 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 The COMPUGRAPH II Working Group

The COMPUGRAPH II Working Group consists of the following team members and European cooperators.

- **Berlin:** *Prof. Dr. H. Ehrig*, Dr. M. Löwe, M. Korff, G. Taentzer, A. Wagner, L. Ribeiro, J. Müller, R. Heckel, R. Betschko **Cooperators:** Prof. Dr. H. J. Schneider, Prof. Dr. F. Parisi-Presicce, Prof. Dr. G. Valiente
- **Bordeaux:** *Prof. Dr. B. Courcelle*, Prof. Dr. Y. Métivier, Prof. Dr. G. Sénizergues, Prof. Dr. Strandh, Dr. M. Billaud, Dr. W. Zielonka, Dr. E. Sopena, Dr. M. Bauderon, Dr. I. Durand, Dr. I. Litovsky, M. Mosbah, A. Pariès, D. Lapoire **Cooperators:** Prof. Dr. Arnborg, Prof. Dr. Dauchet, Prof. Dr. Raoult
- **Bremen:** *Prof. Dr. H.-J. Kreowski*, Dr. B. Hoffmann,Dr. D. Plump, S. Kuske, F. Drewes, R. Klempien-Hinrichs **Cooperator:** Prof. Dr. C. Lautemann, Prof. Dr. A. Habel
- **Brussels/Antwerp (since October 1994):** *Prof. Dr. D. Janssens*, J. Heirbaut, T. Mens **Cooperator:** Prof. Dr. F. Brandenburg
- **Leiden:** *Prof. Dr. G. Rozenberg*, Prof. Dr. G. Engels, Dr. J. Engelfriet, Dr. P. Koopman, M. Andries **Cooperator:** Prof. Dr. M. Nagl
- **Pisa:** *Prof. U. Montanari*, Prof. Dr. P. Degano, Prof. Dr. Maggiolo-Schettini, Dr. A. Corradini, Dr. F. Rossi **Cooperators:** Prof. Dr. R. de Nicola, Prof. Dr. F. Parisi-Presicce

In addition to the European cooperators mentioned above there are close links to other ESPRIT Basic Research Actions, especially COMPASS and SEMAGRAPH, and to international experts in the area of graph transformations all over the world.

1.2 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**

The main research goals and results in these areas will be summarized in the following sections of this paper. For most of the explicit references of the corresponding papers, about 200, published within COMPUGRAPH II, we refer to the first two interim reports [17] and [18]. For a summary of the results of the first phase of COMPUGRAPH we refer to the surveys [14] and [16].

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 main results in the different areas are the following:

2.1 Algebraic Approaches

The algebraic approach 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.

While the foundations of the DPO-approach were almost stable in the first phase of COMPUGRAPH significant new results have been obtained in cooperation of the groups in Pisa and Berlin concerning suitable notions for abstract graph derivations based on standard isomorphisms and typed graph grammars [2].

Concerning the SPO-approach the basic theory has been developed by M. Löwe in his PhD-thesis and subsequent publications in the first phase of COMPUGRAPH [31]. In the second phase significant extensions have been made by the Berlin and Erlangen groups concerning attributed graphs, which can be considered as dynamic abstract data types, and graph transformations with application and consistency conditions [21], which are important in view of specification by graph transformation.

2.2 Categorical Approaches

The theory of high-level-replacement systems, short HLR-systems, is a categorical generalization of the DPO-approach which was developed in the first phase by the groups in Berlin and Bremen in cooperation with F. Parisi-Presicce (Rome) in order to unify DPO-approaches based on different kinds of graphs and structures [11]. In the second phase theory and applications of HLR-systems were extended in various ways, especially to include canonical derivation sequences in the theory [13] and transformations of algebraic specifications and algebraic high-level nets in the applications. Moreover, the HLR-approach was extended to the SPO-approach in the second phase [15] An interesting notion of categorical graph grammars was developed by H. J. Schneider (Erlangen) [37].

2.3 Context Free Graph Grammars and Extensions

Several interesting results concerning unification of context-free and non-context-free approaches for graph grammars have been obtained already by the groups in Bordeaux, Bremen and Leiden in the first phase of COMPUGRAPH. An excellent survey of hyperedge-replacement (HR) is given in the PhD-thesis by A. Habel (Bremen/Hildesheim) [20]. In the second phase the relationship between hyperedge- and vertex-replacement has been studied in more detail by B. Courcelle (Bordeaux) [6] and an interesting extension of HR-grammars, called collage grammars, was studied in the Bremen group, where important decidability results and interesting applications to fractal geometry were obtained [22]. Moreover, the extension of context-free graph grammars using application conditions, mentioned in 2.1, allows to consider rules with positive and negative context conditions which increase their expressive power.

2.4 Logical Approach

In the first phase B. Courcelle has used already monadic second-order logic as a formal language to express graph properties and to characterize graph transductions [5]. The relationship between graph grammars and tree transducers has been studied by J. Engelfriet (Leiden) [19]. In the second phase this was extended to specify also non-context-free sets of graphs and hypergraphs and to define linear orders on graphs with applications to traces and modular decomposition of graphs [7].

2.5 New Approaches (2-Structures and Graph Relabeling)

Two interesting new approaches have been developed in the second phase: The concept of 2-structures, a variant of graphs, developed by G. Rozenberg (Leiden), in cooperation with Boulder and Turku, has been extended to transformations of 2-structures [9] with applications to networks, communication and texts. The concept of graph relabeling systems, developed in Bordeaux, allows to study graph transformations without changing the underlying structure but only the labeling of the graph [30].

3 Concurrency

The main research goal in this area is a high-level concept for concurrent and distributed computation based on graph transformations which is suitable to express various kinds of semantics of concurrent computations as well as specifications of distributed systems based on graph transformations. The main results in this area are the following:

3.1 Truly Concurrent Semantics

In a joint research activity the Berlin and Pisa groups have started to provide graph grammars with a truly concurrent semantics in terms of event structures, similar to the corresponding construction for Petri nets. The ultimate goal for this research is to make graph grammars even a better candidate as model for the specification of concurrent and distributed systems. In particular the Pisa group has developed a notion of "graph processes" generalizing Golz-Reisig net processes [4]. The event structure construction has been obtained for "safe" graph grammars (avoiding automorphisms of graphs) in a first step [3] and was then extended to general "consuming" grammars, i.e. each graph grammar rule deletes at least one item. The event structure construction is based on the construction of abstract graph derivations (2.1) in contrast to a corresponding construction by Schied (Erlangen) based on concrete representations of graphs [38]. It remains open to extend the construction to become an adjunction between the categories of graph grammars and event structures similar to the case of Petri nets.

3.2 Canonical Parallel Derivations

Already in the late 70's H.-J. Kreowski has shown in his PhD-thesis existence and uniqueness of canonical derivations for graph grammars in the DPO-approach with parallel productions. Canonical derivations are the graph grammatical counterpart to leftmost derivations in Chomsky grammars. Within COMPUGRAPH II the groups in Berlin and Bremen were able to generalize this result to HLR-systems (2.2) including graphs, hypergraphs, structures, algebraic specifications and algebraic high-level nets [13]. Moreover, H.-J. Kreowski has presented an axiomatic approach to canonical derivation sequences [27].

3.3 Parallel and Distributed Graph Grammars and Synchronization Mechanisms

Parallel and distributed graph grammars have been developed in the algebraic approach already in the 70's and 80's. Within COMPUGRAPH II a systematic study of different kinds of these graph grammars was done by G. Taentzer (Berlin) showing exactly how these approaches are related to each other and how they are able to express different synchronization mechanisms [41].

3.4 Extended Structure Morphism Systems

Motivated by a notion of Actor grammars [23, 25] which has been developed in Leiden in the late 80's in order to model actor systems by graph grammars the new concept of extended structure morphisms, short ESM-systems, has been developed by D. Janssens within COMPUGRAPH II [24]. In this approach the basic objects, called computation structures, are not states represented by graphs - as usual in other graph transformation approaches - but rewriting processes, while system states can be considered as special cases. In the theory of ESM-systems the communication of processes corresponds to a simple pushout construction. As main results a compositional semantics for ESM-systems is given and it is shown how classical Actor systems and Petri nets as well as new interesting variants of them can be obtained as special cases of ESM-systems.

3.5 Graph Transformations and Other Models of Concurrency

As pointed out in 3.1 and 3.4 already, graph transformations resp. ESM-systems allow to model Actor systems and Petri nets, which are well-known models of concurrency. Actor systems have also been modeled by M. Korff by algebraic actor graph grammars based on equationally defined graph structure transformation in the SPO-approach [26]. In the first phase of COMPUGRAPH it was already shown by the Pisa group how to represent logic programming by context-free graph grammars [8]. This idea was extended in the second phase by modeling concurrent constraint programming using graph transformations in the DPO-approach with context represented by the interface graph [33]. This idea was applied to Petri nets leading to the new notion of contextual nets [34]. Another interesting application of graph transformations to Petri nets is to consider place/transition, colored, or algebraic high-level nets as objects in the sense of high-level replacement systems (2.2) which allows to transform the dynamic structure of different kinds of Petri nets in a rule-based way. It was also shown by the Berlin group that the relationship between graph grammars and place transition nets can be lifted to attributed graph grammars and algebraic high-level nets. Finally it was shown by the Pisa group that state charts can be modeled by graph grammars such that dynamic changes of configurations of state charts correspond to graph transformations [32].

4 Graph Transformation for Specification and Programming

In the first phase of COMPUGRAPH 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. The main results in these areas are the following:

4.1 Term Graph Rewriting

In the first phase of COMPUGRAPH and SEMAGRAPH it has been shown already that 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. Main results of the theory for the evaluation of functional expressions by hypergraph rewriting based on the algebraic DPO-approach are summarized in the PhD-thesis by D. Plump (Bremen) [36]. In the second phase of COMPUGRAPH these results have been extended by the groups in Bremen, Pisa and Berlin taking into account also folding steps for the identification of common subexpressions, memoization, and implementation technique known from function programming, narrowing, which combines conditional term rewriting with unification, graph rewriting, and critical pair analysis in analogy to well-known results in term rewriting [35].

4.2 Graph Rewrite Systems and Languages

In the first phase of COMPUGRAPH the Berlin group has developed the AGG-System, an implementation of algebraic graph rewriting based on the SPO-approach (2.1). In the second phase this AGG-system was documented and applied [29] and a system called BIZARR to visualize collage and other grammars is under development in Bremen.

The graph rewriting language PROGRES has been developed by A. Schürr (Aachen) extending the well-known IPSEN-approach, initiated by M. Nagl and his group in the 80's, for software development using graph transformations based on the algorithmic approach. IPSEN and PROGRES have been successful already in several practical applications, but do not include parallelism and modularity [39].

A new graph and rule centered language, called GRACE, is in an early stage of development in the second phase of COMPUGRAPH by the group in Bremen in cooperation with Aachen, Berlin and Leiden [28]. The main idea of GRACE is that of structured programming with graph transformation units which are independent of a specific graph grammar approach.

4.3 Modularity of Graph Transformation Systems

A first kind of modularity for graph transformation systems is the notion of distributed graph transformations studied in the late 80's for the DPO-approach (2.1) and in the first phase of COMPUGRAPH for the SPO-approach (3.3). Within COMPUGRAPH II modularity was considered to be one of the important aims and different modularity concepts based on distributed graph transformations, the inheritance concept from object-oriented programming, and an import-export-concept similar to algebraic module specifications is under development by the groups in Leiden, Aachen and Berlin [10]. Moreover, transformation units in the sense of GRACE (4.2) can be considered as a modularity concept in its own right or as integrated part of other concepts. An important aspect for modularity is also the integration of graph types and typed graph transformations which are studied by the groups in Leiden and Berlin in cooperation with Aachen. But these are only first attempts in order to obtain suitable modularity concepts for graph transformation systems and languages. Another aspect is the combination of graph transformation systems using categorical methods studied by the groups in Pisa and Berlin.

4.4 Applications to Languages and Software Systems

As pointed out in 4.1 and 4.2 already, graph transformations have been used successfully for specification and implementation of functional languages and software systems before and within the first phase of COMPUGRAPH. Within COMPUGRAPH II several new applications have been studied. The most important language applications are a hybrid query language for databases and a visual language for parsing of pictures by the Leiden group [1], and the semantics of the object-based database language TROLL light by A. Wagner (Berlin) in cooperation with M. Gogolla (Braunschweig/Bremen) [43]. Concerning applications of graph transformations for specification of software systems let us mention main aspects of the specification of the AGG-system using parallel graph transformation systems and of the software development process using different kinds of graph transformations. A "3D-Model of Software Engineering" using PROGRES and a "G2L-System" for the design of modular artificial neural networks was developed by the group in Leiden, a "Character Recognition System" using attributed graph transformations by G. Valiente (Palma de Mallorca), a revision control system [42] and CIM applications by the group in Aachen.

References

[1] M. Andries and G. Engels: Syntax and Semantics of Hybrid Database Languages. In: H. Ehrig and H.J. Schneider (eds.): Graph Transformations in Computer Science. Lecture Notes in Computer Science 776, Springer, 1994, pp. 19–36

[2] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi: Abstract graph derivations in the double pushout approach. In: H. Ehrig and H.J. Schneider (eds.): Graph Transformations in Computer Science. Lecture Notes in Computer Science 776, Springer, 1994, pp. 86–103

[3] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, F. Rossi: An Event Structure Semantics for Safe Graph Grammars. In: Programming, Concepts, Methods and Calculi - PROCOMET'94, IFIP Transactions A - 56, 1994

[4] A. Corradini, U. Montanari, and F. Rossi: Graph processes. Accepted for special issue of Fundamenta Informaticae, 1994

[5] B. Courcelle: Monadic Second-Order Definable Graph Transductions, A Survey. TCS 126 (1994), pp. 53-75

[6] B. Courcelle: Context-free graph grammars: separating vertex replacement from hyperedge replacement. FCT 1993, LNCS 710 (1993), 181-193

[7] B. Courcelle: The Definition in Monadic Second Order Logic of Modular Decomposition of Graphs. To appear in: Proc. 5th Int. Workshop on Graph Grammars, Williamsburg, USA, 1994

[8] A. Corradini and F. Rossi: Hyperedge replacement jungle rewriting for term rewriting systems and logic programming. Theoretical Computer Science, 109:7–48, 1993

[9] A. Ehrenfeucht, G. Rozenberg: Dynamic Labeled 2-Structures. MSCS vol 4 (1994), pp. 433-455

[10] H. Ehrig, G. Engels: Towards a Module Concept for Graph Transformation Systems. Techn. Report Leiden University, No. 93-34, 1993

[11] H. Ehrig, A. Habel, H.-J. Kreowski, F. Parisi-Presicce: Parallelism and concurrency in High Level Replacement Systems. Math. Struct. in Comp. Sci. **1** (1992), 361–404

[12] H. Ehrig,H.-J. Kreowski, G. Rozenberg (eds.): Graph Grammars and Their Applications to Computer Science, Proc. 4th Int. Workshop, Bremen 1990, Springer LNCS 532, (1991)

[13] H. Ehrig, H.-J. Kreowski, and G. Taentzer: Canonical derivations for high-level replacement systems. In: H. Ehrig and H.J. Schneider (eds.): Graph Transformations in Computer Science. Lecture Notes in Computer Science 776, Springer, 1994, pp. 153–169

[14] H. Ehrig, M. Löwe (eds.): Computing by Graph Transformation: Overall Aims and New Results. Springer LNCS 532 (1991), pp. 688-703

[15] H. Ehrig, M. Löwe: Categorical Principles, Techniques and Results for High-Level Replacement Systems in Computer Science, Applied Categorical Structures no. 1, 1993, pp. 21-50

[16] H. Ehrig, M. Löwe (eds.): The ESPRIT Basic Research Working Group COMPUGRAPH "Computing by Graph Transformation", A Survey. TCS 109 (1993), pp. 3-6

[17] H. Ehrig, M. Korff (eds.): Computing by Graph Transformation II, First Interim Report: ESPRIT BR WG COMPUGRAPH. Techn. Report TU Berlin, FB 13, No. 94-4, 1994

[18] H. Ehrig, M. Korff (eds.): Computing by Graph Transformation II, Second Interim Report: ESPRIT BR WG COMPUGRAPH. Techn. Report TU Berlin, FB 13, No. 94-35, 1994

[19] J. Engelfriet: Graph grammars and tree transducers. Proc. CAAP'94, S. Tison (ed.), Lecture Notes in Computer Science 787,1994, pp. 15-36

[20] A. Habel: Hyperedge Replacement Grammars and Languages. Springer LNCS 643, 1992

[21] A. Habel, R. Heckel, and G. Taentzer: Graph grammars with negative application conditions. Accepted for special issue of Fundamenta Informaticae, 1995

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

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

[24] D. Janssens: ESM-Systems and the Composition of their Computations, In: H. Ehrig and H.J. Schneider (eds.): Graph Transformations in Computer Science. Lecture Notes in Computer Science 776, Springer, 1994, pp. 203-217

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

[26] M. Korff: Single pushout transformations of equationally defined graph structures with applications to actor systems. In: H. Ehrig and H.J. Schneider (eds.): Graph Transformations in Computer Science. Lecture Notes in Computer Science 776, Springer, 1994, pp. 234–247

[27] H.-J. Kreowski: An axiomatic approach to canonical derivations. Proceedings 13th IFIP World Computer Congress 1994, Elsevier Science B.V. (North Holland), 1994, pp. 348–353

[28] H.-J. Kreowski: Graph Grammars for Software Specification and Programming: An Eulogy in Praise of GRACE. In: G. Valiente, F. Rosello (eds.): Proc. Coll. on Graph Transf., Tech. Report B - 19 of Univ. of Balerean Islands, 1995

[29] M. Löwe and M. Beyer: AGG — An Implementation of Algebraic Graph Rewriting. *Proc. Fifth Int. Conf. on Rewriting Techniques and Applications*, Springer Lecture Notes in Computer Science 690, 1993

[30] I. Litovsky, Y. Metivier, E. Sopena: Different local Controls for Graph Relabeling System. To appear in Math. Syst. Theory

[31] M. Löwe: Algebraic approach to single pushout graph transformation. Theoret. Comput. Sci. 109, 1993, 181–224

[32] A. Maggiolo-Schettini and A. Peron: Semantics of full Statecharts based on graph rewriting. In: H. Ehrig and H.J. Schneider (eds.): Graph Transformations in Computer Science. Lecture Notes in Computer Science 776, Springer, 1994, pp. 265–279

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

[34] U. Montanari and F. Rossi: Contextual occurrence nets and concurrent constraint programming. In: H. Ehrig and H.J. Schneider (eds.): Graph Transformations in Computer Science. Lecture Notes in Computer Science 776, Springer, 1994, pp. 280–295

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

[36] D. Plump: Evaluation of functional expressions by hypergraph rewriting. Ph.D. Thesis, Universität Bremen, 1993

[37] H.-J. Schneider: On categorical graph grammars integrating structural transformations and operations on labels. Theoretical Computer Science 109, 1993, pp. 257–274

[38] G. Schied: On Relating Rewriting Systems and Graph Grammars to Event Structures. In: H. Ehrig and H.J. Schneider (eds.): Graph Transformations in Computer Science. Lecture Notes in Computer Science 776, Springer, 1994, pp. 326–340

[39] A. Schürr: Logic Based Structure Rewriting Systems. In: H. Ehrig and H.J. Schneider (eds.): Graph Transformations in Computer Science. Lecture Notes in Computer Science 776, Springer, 1994, pp. 341–357

[40] G. Taentzer and M. Beyer: Parallel Graph Transformation Systems and Their Use for Specifying AGG — an Algebraic Graph Grammar System. In: H. Ehrig and H.J. Schneider (eds.): Graph Transformations in Computer Science. Lecture Notes in Computer Science 776, Springer, 1994, pp. 380–394

[41] G. Taentzer: Towards synchronous and asynchronous graph transformations. Accepted for special issue of Fundamenta Informaticae, 1994

[42] B. Westfechtel: Revision and Consistency Control in an Integrated Software Environment (in German). Ph.D. Thesis, RWTH Aachen, IFB 280, Berlin: Springer, 1991

[43] A. Wagner and M. Gogolla: Defining operational behavior of object specifications by attributed graph transformations. Accepted for special issue of Fundamenta Informaticae, 1994