THIRD INTERIM REPORT

ESPRIT Basic Research Working Group No. 7183

Computing by Graph Transformation II (COMPUGRAPH II)

edited by
H. EHRIG and G. TAENTZER

October 1995

Contents

Chapter 1

Introduction and Survey

This report gives an overview over the ongoing activities of the ESPRIT Basic Research Working Group "Computing by Graph Transformation" started on March 1st, 1989. Its second phase —COMPUGRAPH II— began on October 1st, 1992 and lasts to March 31st, 1996. The main aim of the COMPUGRAPH project is to demonstrate the potential of graph transformation as a uniform framework for the development of modern software systems. The research activities therefore cover the whole spectrum from theoretical investigations to practical software engineering applications. This is reflected in the following research areas according to which the work in the COMPUGRAPH project has been organized:

- *Foundations*,
- *Concurrent computing*, and
- *Graph transformation for specification and programming.*

Within the area of *foundations*, research is directed towards the unification, extension, and completion of existing graph transformation theories. In *concurrent computing*, the aim is to elaborate a graph transformation-based approach which can provide operation semantics for various models of concurrent computing (including Petri-nets and process description languages) and enhances correct and comprehensive specifications. The *graph transformation for specification and programming* area considers graph transformation as the kernel of a rule-based specification language and aims at extending this kernel by typing mechanism, module and transformation concepts, applicability constraints, transactions, and concepts developed within concurrent computing in order to improve the feasibility of graph rewriting as a system specification method.

Chapter 2

Summary of the COMPUGRAPH activities

The ESPRIT Basic Research Working Group "Computing by Graph Transformation" started March 1, 1989 and the second phase (COMPUGRAPH II) began on October 1, 1992.

Due to its character as a working group, the project supports the cooperation of the partners. Research on graph transformation was intensified by several bilateral meetings of the partners, by guest lectures of the European cooperators and international experts, and by some joint work of the project teams.

2.1 Scientific Progress

Research in the first year of the project has followed the outline which was described in the proposal for COMPUGRAPH II and the technical annex of the contract. The progress w.r.t. these research issues and the results obtained up to now are described in chapter 3. Most of the results have been published already in scientific journals or proceedings, compare chapter 5.

2.2 Workshops

In the previous year of the project two workshops have been organized and executed by partners of the COMPUGRAPH II-project.

5th Int. Gragra-Workshop, USA, Nov 94: This workshop organized by J. Cuny (USA) in cooperation with H. Ehrig, G. Engels, and G. Rozenberg took place on November 13–18, 1994. The workshop has been organized on invitation basis together with an additional open call for papers. On account of a joint NSF/ESPRIT-proposal successfully worked out by the organizers 50.000 ECU have been available as financial support for European participants. The proceedings of this workshop will be published in Springer LNCS.

SEGRAGRA-Workshop, Volterra, Italien, Aug 1995: This workshop on Graph Rewriting and Computation was a joint activity of the two Esprit Basic Research Working Groups, COMPUGRAPH II and SEMAGRAPH II organized by A Corradini and U.

Montanari. It was the final workshop for both projects where a coherent, detailed account of achieved results was presented. Proceedings will be published in Electronic Notes of Theoretical Computer Science (a new series by Elsevier associated with the TCS journal and edited by M. Mislove, M. Nivat and C. Papadimitriou).

2.3 Cooperation with other ESPRIT Basic Research Projects

Cooperation with the ESPRIT Basic Research Actions "COMPASS II", "ISCORE II", "ASMICS II", and "SEMAGRAPH II"(see section 2.2 above) has fruitfully been continued and hence extended the previous cooperations within the first phases of these projects.

2.4 The COMPUGRAPH II Working Group

The COMPUGRAPH II Working Group consists of the following team members (team leaders are printed in italics) and European cooperators.

Berlin: *Prof. Dr. H. Ehrig*, Dr. M. Löwe, M. Korff, G. Taentzer, A. Wagner, J. Padberg, 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. R. 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*, Joris Heirbaut, Tom 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
Cooperator: Prof. Dr. R. de Nicola, Prof. Dr. F. Parisi-Presicce

Chapter 3

Progress Report

The structure of the progress report corresponds to the structure of the research areas and topics given in the proposal for COMPUGRAPH II and technical annex of the contract. Most of the results have been published already in scientific journals or proceedings (compare the references).

3.1 Foundations

The research in the area of foundations can be structured according to the following subareas.

3.1.1 Algebraic Approach to Graph Transformation

In the single-pushout approach to graph transformation a new categorical framework of so-called graph-interpreted graph transformation has been presented in [Kor94]. Single pushout transformation allowing partial graphs has been introduced in [WG95].

Graph grammars with application conditions have been studied in [Hec95, HHT95, HW95, Wag95]. In [HHT95] these kinds of graph grammars have been introduced and the focus was put on comparing their generative power with those of the classical approaches. Their embedding properties have been studied in [Hec95]. 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.

The theory of high-level replacement systems has been further extended by results about embedding of derivations in [PT95].

3.1.2 Structuring of Graph Transformation

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+95] 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.

3.1.3 Context-free graph grammars and graph transductions

Drewes, Habel, and Kreowski [DHK95] have written a chapter on hyperedge replacement for the Handbook of Graph Transformations.

Klempien-Hinrichs [Kle95] introduces node replacement in hypergraphs as a generalization of node replacement in graphs, yielding hypergraph grammars with neighbourhood controlled embedding (hNCE-grammars). This approach can simulate hyperedge replacement, where the constructed grammars are boundary. The main result of [Kle95] is that confluence is a decidable property of hNCE-grammars. This is proved by a triply exponential-time decision algorithm.

Collage grammars provide devices for the generation of pictures by adapting techniques and ideas of hyperedge replacement graph grammars. In [DHT95], Dassow, Habel, and Taubenberger show 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.

Drewes [Dre95a] investigates the power of top-down, bottom-up, and tree-to-graph-to-tree transducers (tgt transducers) to compute translations from hyperedge-replacement algebras into edge-replacement algebras. 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 tgt transducers for the computation of functions into a semiring.

Courcelle has written a tutorial survey on the use of universal algebra in the area of graph grammars [Cou95a], he has written a chapter on monadic second-order logic for the Handbook of Graph Transformations [Cou95b], and he has given tutorial lectures on these subjects [Cou95c, Cou95d].

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

Rozenberg has written a chapter on 2-structures for the Handbook of Graph Transformations (with co-authors outside COMPUGRAPH) [EHR95]. Engelfriet and Rozenberg have written a chapter on node replacement graph grammars (in particular C-edNCE grammars) for the Handbook of Graph Transformations [ER95]. Engelfriet has investigated a notion of concatenation of graphs, which allows string languages to be interpreted as graph languages [EV95a, EV95b].

3.1.4 Graph rewriting systems

A concept of graph unification and matching is introduced by Plump and Habel [PH95]. 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 σ such that $L\sigma \cong G$ and (2) applying σ to R. Substitution-based graph rewriting generalizes the double-pushout approach in that rewrite steps can copy or delete non-local subgraphs.

Local graph rewriting systems are studied by Litovsky, Métivier, and Sopena. Their work is a contribution to the study of the general problem of characterizing those properties which can be computed on a graph or on a network by means of local transformations of states (or labels) attached to vertices or edges [LMZ95, LMS95].

Durand uses graphs to implement term rewriting systems [DS94, Dur94].

3.2 Concurrency

The research activity in the area of concurrency during the third year continued in the following subareas.

3.2.1 Truly Concurrent Semantics of Graph Transformation

The joint research activity of the Berlin and Pisa groups aimed at providing graph grammars with a truly concurrent semantics in terms of Event Structures continued in various directions. In [CEL+95] a functorial semantics for so-called safe graph grammars using prime algebraic domains and event structures is introduced. In [CEL+94b] the event structure semantics is further developed for consumptive graph grammars with parallel productions.

Motivated by a corresponding application to actor systems, an event structure semantics is also proposed in [Kor94] for the single pushout approach. This kind of semantics is based on concurrent derivations which can be considered as graph grammar processes ([KR95a]).

A new concept of process for transforming graphs has been proposed in [MW95a]. It should cover previous concepts, including that in [MW82].

3.2.2 Distributed Graph Transformation and Synchronization Mechanisms

In [Tae95] synchronous and asynchronous graph transformations are considered on the basis of distributed transformations in the SPO-approach. Some criteria are given there preserving 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 [Tae95a]. This kind of transformation offers the possibility of modeling different aspects of distributed systems. The topological structure of the system as well as local parts can be described in a graphical way. Different kinds of distributed actions such as local actions, network activities, communication and synchronization can be modeled homogeneously using the same method: graph transformation. This new approach to graph transformation fits into the framework of high-level replacement systems.

A new synchronization operation on productions is introduced in [CR95], 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.

3.2.3 Extended Structure Morphism systems

Actor grammars, which were introduced in [JR89] as a graph rewriting model for actor systems, have been generalized to ESM systems, in [Jan93b], by lifting a number of restrictions on the form of productions. ESM graph rewriting is quite different from graph rewriting in the algebraic approach, because the embedding mechanism, which is more or less trivial in algebraic graph rewriting, plays a central role. Research about ESM systems has focused on the development of a truly concurrent semantics, and on the properties of that semantics with respect to system composition and abstraction.

Since a graph rewriting system is a set of productions, in this framework set union is a natural way to compose systems. Evidently, system composition is essential if one ever wants to represent realistic systems. In [JLR93] and [Jan93b] a process semantics of ESM systems is introduced, and in [Jan93b] and [Jan95] it is shown that the notion of a process (called computation structure in [Jan93b]) can be refined so that the resulting semantics is compositional with respect to the union of sets of productions. The basic idea is that each process is equipped with a condition on the context in which it may occur; if one views a process as a generalized production, then this condition generalizes the notion of the left-hand side of a production. The composition of the semantics objects, which are sets of processes, can be defined elementwise: each process of the composed system can be decomposed into interacting processes of the component systems.

In [JM95] it is demonstrated that one can obtain, for a given system, a truly concurrent semantics at various level of abstraction. This is achieved by starting from a basic process semantics, and then forgetting part of the information in that basic semantics, yielding a more abstract view of the system behaviour. It is shown that these mechanisms are compatible with the composition of computation structures. It is also shown that, in each case, one gets a category where graphs are objects and computation processes are arrows.

As far as the relationship with Petri nets is concerned, it is demonstrated in [JM95] and [Jan95] that one obtains ESM systems from Petri nets by generalizing markings to graph structures, where edge labels are seen as unary operations on vertices. Finally, it turns out that one may easily express various variants of Petri nets by putting restrictions on the set of conditional computations that one considers to be valid. In this way one may, e.g., remove self-concurrency or give places bounded capacity. The semantics obtained are still compositional with respect to the union of sets of productions; when translated into the usual representation of Petri nets, this union corresponds to the gluing of nets over common places.

3.2.4 Modeling Actor Systems and Petri Nets

The Berlin group continued to study the modeling of Actor Systems and of Petri Nets, using algebraic graph grammars. In [Kor94] a true concurrency semantics particularly addressing the fairness requirement of finite delay is given to an algebraic graph grammar model of actor systems. The relationship between algebraic high-level nets and attributed graph grammars is revealed in [KR94, KR95b].

The tutorial paper [Cor95] investigates 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. The paper describes to what extent the truly concurrent semantics of nets has been generalized to graph grammars, and what remains to be done.

3.3 Graph Transformation for Specification and Programming

Research in this area can be roughly divided into the five subareas *Languages Based on Graph Transformation*, *Properties of Graph Programs*, *Term Graph Rewriting*, *Application to Language Definitions* and *Various Applications*.

3.3.1 Languages Based on Graph Transformation

During the last year a final version of the paper on a language for rewriting hypergraphs in a controlled manner has been prepared by Maggiolo-Schettini and Winkowski [MW95]. The language is an evolution of the one in [MW92]. Programs are built from productions by means of constructors which specify order and modalities of the rewriting steps. Programs may contain parameters to point to specific elements to which they are supposed to be applied, and are intended to describe how a given graph and a valuation of parameters in this graph are transformed to obtain a resulting graph and a resulting valuation of parameters. The language has an operational and a denotational semantics.

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 [Tae95a]. The concurrently interacting objects support information hiding and communicate by import-export interfaces.

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.

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.

3.3.2 Properties of Graph Programs

A necessary and sufficient condition for termination of graph rewriting systems is established by Plump [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.

In order to decide confluence, a critical pair analysis for single-pushout graph rewriting is provided by Löwe and Müller [LM95].

3.3.3 Term Graph Rewriting

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, van Eekelen, and Plasmeijer [KEP95]. In order to speed up the evaluation, Zweije and Koopman use symbolic reduction to fully automatically transform term graph rewriting systems into semantically equivalent systems that are operationally more efficient.

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 equivalent to "collapsed tree rewriting" and hence is complete for equational deduction. For term graph narrowing, a completeness result is established which corresponds to Hullot's classical result for term narrowing. 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.

Kuske [Kus95] shows how to implement the β-reduction of the λ-calculus by hypergraph rewriting according to the double-pushout approach. The resulting model, called *collapsed λ-tree rewriting*, is shown to be sound with respect to β-reduction and complete with respect to the Gross-Knuth strategy (which reduces all redexes in a λ-term in parallel). As a consequence, a collapsed λ-tree is normalizable by collapsed λ-tree rewriting if and only if the represented term is normalizable by β-reduction.

3.3.4 Application to Language Definitions

Wagner and Gogolla [WG95] 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 [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. They 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].

3.3.5 Various Applications

In [MP94a], Maggiolo-Schettini and Peron propose graph rewriting as a proper "abstract machine" for defining an operational semantics for the specification formalism Statecharts. A statechart is translated into a set of graph productions and steps from configuration to configuration of a statechart are described in terms of a derivations. So the sequential behaviour of the represented statechart is described by the derivation graph. In [MP94b] 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.

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 [Bet94], Betschko uses graph transformation to support the software development process.

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 [Boe94] was developed by Boers. 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 possible to encode these changes into the production rules that generated the initial topology, a Lamarckian approach seems not to be possible.

Chapter 4

Planned Activities

The work plan for the last half year of the project "Computing by Graph Transformation" is going to follow the guidelines which have been agreed on in the technical annex of the contract.

4.1 Research

As it has been exhibited in section 2, research in the third year followed the guidelines of the proposal for COMPUGRAPH II and the technical annex of the contract. All areas have been studied and the effect of the project has been a closer cooperation of the partners which led to intensive mutual research exchange (lectures, visits, etc.), joint papers, and many results w.r.t. unification of research directions.

Research in the last half year is going to continue these research lines. The papers in preparation shall be finished within the remaining time.

4.2 Deliverables

A survey of research goals and main results of the COMPUGRAPH II project was given in [ET95]. Two main activities are concerned with the survey and presentation of the overall concepts and results in the COMPUGRAPH II-project.

Final Report and Annotated Bibliography: Whereas the final report should contain the new contributions made in the COMPUGRAPH II period only, the annotated bibliography should consider the whole time graph transformation has been studied. The long period before COMPUGRAPH II should be described along few key contributions in the three main areas of graph transformation considered in COMPUGRAPH II, particularly emphasizing the contributions of the COMPUGRAPH I project. These contributions should be selected mainly from published papers. Very important other papers are also possible.

A list of all references mentioned in the final report of COMPUGRAPH I is send from Berlin to all partners by October/November this year. The partners will select those contributions which seem to be important for them and describe them shortly. Similarly to the reports the annotated bibliography will also be structured according to the areas of COMPUGRAPH II. Moreover, it is intended to publish the annotated bibliography in the EATCS bulletin.

Handbook Publication Proposal: A "Handbook of Graph Transformations by COMPUGRAPH" is planned which consists of three parts according to the overall structure of the project (Foundations, Concurrency and Specification and Programming). The first part "Foundations" shall collect nearly all different approaches to graph transformation each presented on ca. 40 - 50 pages by different authors. By now, all chapters of the handbook are written and as far as he remembers all referee reports have reached Leiden. Editing should be finished at the end of this year and a final layout should be done to adapt the chapters to a common style.

Chapter 5

References

[AE95a] Marc Andries and Gregor 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] Marc Andries and Gregor Engels. A Hybrid Query Language for the Extended Entity Relationship Model. *Journal of Visual Languages and Computing*, 1995. To appear.

[AFHS95] Marc Andries, Rudolf Freund, Brigitte Haberstroh, and Christian 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] Marc Andries, Rudolf Freund, and Christian 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] Marc Andries. *Graph Grammars for Visual Database Languages.* PhD thesis, Leiden University, The Netherlands, February 1996.

[Bau95a] M. Bauderon; A uniform approach to graph rewriting: the pullback approach, *Proc. WG'95*, Aachen, June 20-23, 1995, to appear

[Bau95b] M. Bauderon; Parallel rewriting of graphs through the pullback approach, *Proc. SEGRAGRA'95*, Volterra, September 1995, *Elect. Notes in Theor. Comp. Sci.*

[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.

[Bet94] 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.

[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.

[CEL+95] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and J. Padberg. Typed graph grammars and their adjunction with categories of derivations. Extended abstract, submitted for Proceedings of Graph Grammar Workshop, Williamsburg, 1994.

[CEL+94b] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. An event structure semantics for graph grammars with parall productions. To appear in *Proc. of 5. Int. Workshop on Graph Grammars and Their Applications to Computer Science*, Lecture Notes in Computer Science, (Springer, Berlin), 1995.

[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.

[CH95] A. Corradini and R. Heckel. A compositional approach to structuring and refinement of typed graph grammars. To appear in Proc. of SEGRAGRA'95 "Graph Rewriting and Computation", Electronic Notes of TCS, 1995.

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

[Cor95] A. Corradini, *Concurrent Computing: From Petri Nets to Graph Grammars*, to appear in *Proceedings of the Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation*, Volume 2 of Electronic Notes in Theoretical Computer Science, Elsevier, 1995.

[Cou95a] B. Courcelle: Basic notions of universal algebra for language theory and graph grammars, Theoret. Computer Science (Tutorial section), to appear

[Cou95b] B. Courcelle: The expression of graph properties and graph transformations in monadic second-order logic, 1995, Chapter for the Handbook of Graph Transformations, Volume I: Foundations (G.Rozenberg, ed.), World Scientific, to appear

[Cou95c] B. Courcelle: Logic and graphs, Tutorial delivered at the Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation, Volterra, Italy, Aug. 1995

[Cou95d] B. Courcelle: The construction of obstructions from graph grammars, 3rd Slovenian International Conference in Graph Theory, Bled, Slovenia, June 25-30, 1995

[CR95] A. Corradini and F. Rossi, *Synchronized Composition of Graph Grammar Productions*, to appear in *Proceedings fifth International Workshop on Graph Grammars and their Applications to Computer Science*, Lecture Notes in Computer Science, Vol. ??? (Springer, Berlin, 1995).

[DHK95] Frank Drewes, Annegret Habel, Hans-Jörg Kreowski. Hyperedge replacement. Chapter for the Handbook of Graph Transformations, Volume I: Foundations (G.Rozenberg, ed.). World Scientific. To appear.

[DHT95] Jürgen Dassow, Annegret Habel, Stefan 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.

[Dre95a] Frank 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] Frank Drewes. Semirings and tree-to-graph-to-tree transductions. In *Proc. SEGRAGRA '95*, Electronic Notes in Theoretical Computer Science. Elsevier, 1995. To appear.

[DS94] Irène Durand, Bruno Salinier; Constructor equivalent term rewriting systems are strongly sequential: a direct proof, Information Processing Letters 52 (1994), 137–145

[Dur94] Irène Durand; Bounded, strongly sequential and forward-branching term rewriting systems, Journal of Symbolic Computation 18 (1994), 319–352

[EE94] H. Ehrig and G. Engels. Pragmatic and semantic aspects of a module concept for graph transformation systems. Submitted for *Proc. International Workshop on Graph Grammars and Their Application to Computer Science '94*.

[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.

[EHR95] A.Ehrenfeucht, T.Harju, G.Rozenberg; 2-Structures, Chapter for the Handbook of Graph Transformations, Volume I: Foundations (G.Rozenberg, ed.), World Scientific, to appear

[ER95] J.Engelfriet, G.Rozenberg; Node replacement graph grammars, Chapter for the Handbook of Graph Transformations, Volume I: Foundations (G.Rozenberg, ed.), World Scientific, to appear

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

[ES95b] Gregor Engels and Andy 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. To appear.

[ET95] H. Ehrig and G. Taentzer. COMPUGRAPH II: A survey of research goals and main results. To appear *EATCS Bulletin*.

[EV95b] J.Engelfriet, J.J.Vereijken; Context-free graph grammars and concatenation of graphs, Tech. Report 95-27, Leiden University, 1995

[EV95a] J.Engelfriet, J.J.Vereijken; Concatenation of graphs, in Proc. International Workshop on Graph Grammars and Their Application to Computer Science, Lecture Notes in Computer Science, Springer-Verlag, to appear

[Hec95] R. Heckel. Embedding of conditional graph transformation. In G. Valiente Feruglio and F. Rosello Llompart, eds., *Proc. Colloquium on Graph Transformation and its Application in Computer Science*. Technical Report B-19, Universitat de les Illes Balears, 1995.

[HHT95] A. Habel, R. Heckel, and G. Taentzer. Graph grammars with negative application conditions. Accepted for special issue of Fundamenta Informaticae, 1995.

[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.

[HP95] Annegret Habel and Detlef 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. To appear.

[HW95] R. Heckel and A. Wagner. Consistency of conditional graph grammars. To appear in Proc. of SEGRAGRA'95 "Graph Rewriting and Computation", Electronic Notes of TCS, 1995.

[Jan93a] D. Janssens. Equivalence of computations in actor grammars. *Theoretical Computer Science*, 109:145–180, 1993.

[Jan93b] D. Janssens. ESM systems and the composition of their computations. In *Graph Transformations in Computer Science, LNCS 776*, pages 203–217. Springer-Verlag, 1993.

[Jan95] D. Janssens, The Decomposition of ESM Computations, to appear.

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

[JM95] D. Janssens and T. Mens. Abstract semantics for ESM systems. UIA report 95-04, University of Antwerp.

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

[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] Hans-Jörg Kreowski and Sabine 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.

[Kle95] Renate Klempien-Hinrichs. Node replacement in hypergraphs: simulation of hyperedge replacement, and decidability of confluence. In *Proc. International Workshop on Graph Grammars and Their Application to Computer Science '94*, Lecture Notes in Computer Science. Springer-Verlag. To appear.

[Kor94a] 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 – Corretness and Reusability IS-CORE'94*, pages 244–258, 1994. Tech. Report IR-357, Free University, Amsterdam.

[Kor94b] 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.

[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. To appear in *Proc. of the Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation*, Volume 2 of Electronic Notes in Theoretical Computer Science, Elsevier, pages 113–122, 1995.

[KR95b] M. Korff and L. Ribeiro. Formal relationship between graph grammars and Petri nets. Submitted for the proceedings of the 5th International Workshop on Graph Grammars and their Applications to Computer Science, 1995.

[Kus95] Sabine Kuske. Implementing β-reduction by hypergraph rewriting. 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. To appear.

[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.

[LMS95] I. Litovsky, Y. Métivier, E. Sopena; Checking global graph properties by means of local computations: the majority problem, Proc. SEGRAGRA'95, Electronic Notes in Theoretical Computer Science, 1995, to appear

[LMZ95] I. Litovsky, Y. Métivier, W. Zielonka; On the recognition of families of graphs with local computations, Information and Computation 118 (1995), 110–119

[MP94a] A. Maggiolo-Schettini, A. Peron. Semantics of Full Statecharts Based on Graph Rewriting. Lecture Notes in Computer Science 776, 265–279, Springer-Verlag, 1994.

[MP94b] A. Maggiolo-Schettini, A. Peron. A Graph Rewriting Framework for Statecharts Semantics. *Preliminary Proc. International Workshop on Graph Grammars and Their Application to Computer Science '94*, 1994.

[MW82] Maggiolo-Schettini, A., Winkowski, J., Processes of Transforming Structures, J. Comput. System Sci. 24 (1982) 431–445.

[MW92] A. Maggiolo-Schettini, J. Winkowski. A Programming Language for Deriving Hypergraphs. Lecture Notes in Computer Science 581, 221–231, Springer-Verlag, 1992.

[MW95] A. Maggiolo-Schettini, J. Winkowski. A Kernel Language for Programmed Rewriting of (Hyper)Graphs. *Acta Informatica.* To appear.

[MW95a] Maggiolo-Schettini, A., Winkowski, J., Dynamic Graphs, in preparation.

[PH95] Detlef Plump and Annegret 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.

[Plu95] Detlef 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. Springer-Verlag. To appear.

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

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

[RS94] Jan Rekers and Andy 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] Jan Rekers and Andy 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] Jan Rekers and Andy 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`.

[Tae95] G. Taentzer. Towards synchronous and asynchronous graph transformations. Accepted for special issue of Fundamenta Informaticae, 1995.

[Tae95a] G. Taentzer. Hierarchically distributed graph transformation. To appear in *Proc. of 5. Int. Workshop on Graph Grammars and Their Applications to Computer Science*, Lecture Notes in Computer Science, (Springer, Berlin), 1995.

[TS95] G. Taentzer and A. Schürr. DIEGO, another step towards a module concept for graph transformation systems. 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. To appear.

[Wag95] A. Wagner. On the expressive power of graph grammars with application conditions. In *TAPSOFT/CAAP*, 1995.

[WG95] A. Wagner and M. Gogolla. Defining operational behavior of object specifications by attributed graph transformations. Accepted for special issue of Fundamenta Informaticae, 1995.

[***] **Updated references of the second interim report:**

[Bau94] M. Bauderon; A categorical approach to vertex replacement: the generation of infinite graphs, 5th International Conference on Graph Grammars and Their Application to Computer Science, Williamsburg, November 1994, to appear

[BE94] H.L.Bodlaender, J.Engelfriet; Domino treewidth, in Proc. Graph Theoretic Concepts in Computer Science WG'94, Lecture Notes in Computer Science 903, Springer, 1995, pp.1–13; also Tech. Report UU-CS-1994-11, Utrecht University

[Cou94a] B.Courcelle: Monadic second-order logic and linear orderings of finite structures, 8th Workshop on Computer Science Logic, Kazimierz, Poland, Sept. 94, Proceedings: LNCS, Volume 933, 1995, 163-176

[CS94] B. Courcelle, G. Senizergues: The obstructions of a minor-closed set of graphs defined by a context-free grammar, 1995, submitted to Discrete Maths.

[DHKT94] Frank Drewes, Annegret Habel, Hans-Jörg Kreowski, Stefan Taubenberger. Generating self-affine fractals by collage grammars. *Theoretical Computer Science* 145, 159–187, 1995.

[***] **Updated references of the first interim report:**

[CMet93] B. Courcelle, Y. Métivier, Coverings and minors: application to local computations in graphs, European Journal of Combinatorics 15 (1994), 127–138

[EHPR93] J.Engelfriet, T.Harju, A.Proskurowski, G.Rozenberg; Characterization and complexity of uniformly non-primitive labeled 2-structures, Report 94-31, Leiden University, to appear in TCS

[EV92] J.Engelfriet, H.Vogler; The translation power of top-down tree-to-graph transducers, JCSS 49 (1995), 41–65

[LMS93] I. Litovsky, Y. Métivier, E. Sopena; Different local controls for graph relabelling systems, Math. Syst. Theory 28 (1995), 41–65

[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`).

[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`).