Appligraph

Applications of Graph Transformation

ESPRIT Working Group No. 22565

Fourth Annual Progress Report

Hans-Jörg Kreowski and Detlef Plump
(editors)

May 2001

Contents

1 Introduction

This report summarizes the activities in the fourth year of the ESPRIT Working Group APPLIGRAPH, covering the period from April 1, 2000, to March 31, 2001. The principal objective of this Working Group is to promote applied graph transformation as a rule-based framework for the specification and development of systems, languages, and tools and to improve the awareness of its industrial relevance.

The work in APPLIGRAPH is organized according to the following three research areas:

- *Language Issues* (coordinated by Gregor Engels)
- *Tool Issues* (coordinated by Manfred Nagl)
- *Application Domains* (coordinated by Francesco Parisi-Presicce)

In the area of *Language Issues*, features in graph transformation based languages are investigated, like typing, modularity, refinement, parallelism, concurrency, distribution, optimisation, and correctness.

The area of *Tool Issues* is concerned with the conception and design of support tools for graph transformation based languages including editors, parsers, interpreters, compilers, optimisers, verifiers, and graphical user interfaces.

In *Application Domains*, the usefulness of graph transformation is demonstrated by case studies in various areas like definition of visual languages, database models, concurrent and distributed systems, software process modelling, and implementation of programming languages.

2 Members and Scientific Correspondents

The following research teams participate in APPLIGRAPH:

Aachen University of Technology: Manfred Nagl (team leader), Bernhard Westfechtel, André Marburger, Boris Boehlen, Ansgar Schleicher, Dirk Jäger, Oliver Meyer, Felix Gatzemeier, Manfred Münch, Katja Cremer (until May 2000), Anita Behle (until March 2000), Peter Klein (until June 2000), Ansgar Radermacher (until April 1999), Stefan Gruner (until July 1999), Roland Baumann (until April 1999)

University of Antwerp: Dirk Janssens, Jan Paredaens (team leaders), Jan Van den Bussche, Nico Verlinden

Technical University of Berlin: Hartmut Ehrig, Herbert Weber (team leaders), Martin Grosse-Rhode, Julia Padberg, Gabi Taentzer, Roswitha Bardohl, Claudia Ermel, Thorsten Schultzke, Olga Runge, Maike Gajewsky, Felix Bübl, Michael Matz, Ingo Weinhold, Marta Simeoni, Aliki Tsiolakis, Magnus Niemann

University of Bremen: Otthein Herzog (team leader), Hans-Jörg Kreowski (team leader and Working Group coordinator), Detlef Plump (until September 2000), Frank Drewes (until April 2000), Sigrid Ewert, Martin Gogolla, Berthold Hoffmann, Gregor Joeris, Renate Klempien-Hinrichs, Peter Knirsch, Sabine Kuske

University of Leiden: Grzegorz Rozenberg (team leader), Egbert Boers, Joost Engelfriet, Jetty Kleijn, Vincent Zweije, Sebastian Maneth

University of Nijmegen: Rinus Plasmeijer (team leader), Marko van Eekelen, Sjaak Smetsers, Erik Barendsen, John van Groningen, Peter Achten, Marco Pil, Pascal Serrarens, Ronny Wichers Schreur, Arjan van Yzendoorn, Pieter Koopman, Thorsten Zörner, Maarten de Mol

University of Paderborn:[1] Gregor Engels (team leader), Ralph Depke, Reiko Heckel, Jochen Küster, Katharina Mehner, Stefan Sauer, Annika Wagner

University of Pisa: Ugo Montanari (team leader), Paolo Baldan, Stefano Bistarelli, Roberto Bruni, Andrea Corradini, Andrea Maggiolo-Schettini, Emilio Tuosto

University of Rome: Francesco Parisi-Presicce (team leader), Stefano Levialdi, Piero Mussio, Luigi Cinque, Paolo Bottoni, Marta Simeoni

APPLIGRAPH corresponds and cooperates with other experts in the field and related applied areas. The list of scientific correspondents includes: Virginio Cantoni (Pavia, Italy), Maria Francesca Costabile (Bari, Italy), Heiko Dörr (Daimler-Benz AG, Berlin, Germany), Jürgen Ebert (Koblenz, Germany), Michael Gödicke (Essen, Germany), Herbert Göttler (Mainz, Germany), Marc Gyssens (Limburg, Belgium), Annegret Habel (Oldenburg, Germany), T. Harju (Turku, Finland), Neil Jones (Copenhagen, Denmark), Simon Peyton Jones (Glasgow, Scotland), Michael Löwe (Hannover, Germany), Alfonso Pierantonio (L'Aquila, Italy), Wilhelm Schäfer (Paderborn, Germany), Hans-Jürgen Schneider (Erlangen, Germany), Andy Schürr (München, Germany), Gabriel Valiente Feruglio (Barcelona, Spain), Albert Zündorf (Paderborn, Germany)

[1] University of Paderborn—whose team leader Gregor Engels had moved from Leiden to Paderborn—became a new site of APPLIGRAPH in February 1999

3 Summary of Activities

3.1 Research

During the fourth year of APPLIGRAPH, research has been carried out according to the Working Group Programme as specified in Annex I of the contract.

The results of this research are documented in 7 papers published or accepted for publication in journals and books, 63 papers published or accepted for publication in conference and workshop proceedings, one doctoral dissertation and 10 technical reports. See also Subsection 3.4 for special journal issues and proceedings of workshops in the fourth year. The scientific results are described in Section 4 and references to publications are given at the end of this report.

3.2 Meetings

The international workshops GTVMT 2000 and UniGra 2001, two Subgroup Meetings and a Team Leader Meeting have been organized during the fourth year of the Working Group.

GT-VMT 2000 – Workshop on Graph Transformation and Visual Modeling Techniques.
This event was co-organized by APPLIGRAPH and took place in Geneva (Switzerland) on July 14 and 15, 2000, as a satellite event of the ICALP 2000 conference.

UniGra 2001 – Workshop on Uniform Approaches to Graphical Process Specification Techniques.
This Workshop was co-organized by APPLIGRAPH as well. It was held in Genova (Italy) on March 31 and April 1, 2001, as a satellite event of the ETAPS 2001 joint conference.

Subgroup Meeting in Paderborn.
This meeting took place September 5–6, 2000, at the University of Paderborn (Germany). Its purpose was to discuss common XML-based exchange formats for graphs and graph transformation systems.

Subgroup Meeting in Bremen.
This meeting was held March 1-2, 2001, at the University of Bremen (Germany). It continued the discussion and development on XML-based exchange formats for graphs and graph transformation systems.

Sixth Team Leader Meeting.
This meetings was held at ETAPS 2001 in Genova (Italy), as part of the

meeting of the Steering Committee of the International Conference on Graph Transformation (ICGT). Administrative and financial issues of the Working Group were briefly discussed.

3.3 Future Events

The following events are planned for the final 13 months of the APPLIGRAPH Working Group (April 2001 to April 2002):

GT-VMT 2001 – Second International Workshop on Graph Transformation and Visual Modeling Techniques.
This workshop is co-organized by APPLIGRAPH and will take place on Crete (Greece) on July 12-13, 2001, as a satellite event of the ICALP 2001 conference.

Subgroup Meeting on XML-based exchange formats.
This meeting is scheduled to take place on September 20-21, 2001, at the University of Erlangen (Germany).

AGT 2002 – Final APPLIGRAPH Workshop on Applied Graph Transformation.
This final event of the Working Group will be an open satellite workshop of the ETAPS 2002 joint conference, to be held in Grenoble (France) from April 6 to April 14, 2002.

3.4 Information Dissemination

In the fourth year of APPLIGRAPH, the Working Group continued its efforts to promote research activities in applied graph transformation and to improve the systematic exchange and dissemination of information on graph transformation and its applications. This includes the following activities:

Special Journal Issues on GRATRA 2000. Selected papers of the Joint APPLIGRAPH/GETGRATS Workshop on Graph Transformation Systems (GRATRA 2000) will appear in special issues of the journals *Mathematical Structures in Computer Science* (Cambridge University Press) and *Science of Computer Programming* (Elsevier Publishers). The proceedings of the workshop appeared as a technical report of the Technical University of Berlin, see [ET00b] and `http://tfs.cs.tu-berlin.de/gratra2000`.

Proceedings of GT-VMT 2000. Extended abstracts of the papers presented at the Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2000) have been published by Carleton Scientific in a volume collecting the contributions to all satellite workshops of ICALP 2000.

Proceedings of UniGra 2001. The proceedings of the Workshop on Uniform Approaches to Graphical Process Specification Techniques (UniGra 2001) will appear in a special issue of *Electronic Notes in Theoretical Computer Science* (Elsevier Publishers).

APPLIGRAPH NEWS. An *Electronic Bulletin Board* is used by APPLIGRAPH members to distribute paper announcements and related news via the Internet. This service is based on the GRAGRA mailing list maintained by the Aachen group. The e-mail address for distributing news is `gragra@bach.informatik.rwth-aachen.de`.

WWW Homepage. The Working Group's WWW home page (`http://www.informatik.uni-bremen.de/theorie/appligraph/home.html`) contains information about the project (synopsis, progress reports and proposal) and links to related information (workshops, graph transformation tools, graph grammar bibliography and graph grammar mailing list).

3.5 Further Activities

To establish a sustained promotion of the area of graph transformation, APPLIGRAPH supports the Steering Committee of the International Conference on Graph Transformation (ICGT). Its members are Michel Bauderon, Andrea Corradini, Hartmut Ehrig (chair), Gregor Engels, Dirk Janssens, Hans-Jörg Kreowski, Ugo Montanari, Manfred Nagl, Francesco Parisi-Presicce, John Pfaltz, Rinus Plasmeijer, Azriel Rosenfeld and Grzegorz Rozenberg.

4 Scientific Results of the Fourth Year

The following description of the fourth year's scientific achievements is structured according to three research areas of the Working Group Programme: *Language Issues*, *Tool Issues*, and *Application Domains*.

4.1 Language Issues

In this area topics in graph transformation based languages are investigated, like typing, modularity, refinement, parallelism, concurrency, distribution, optimisation, and correctness.

4.1.1 Modularity

The papers [DKKK00] and [KK00a], which were already described in the Third Annual Report, have now appeared.

Transformation units and transformation modules are known as structuring concepts for graph transformation systems. Their semantics is based on the interleaving of rule applications and calls to imported units. In [KK00b] they are generalized in such a way that they can be used for the structuring of arbitrary rule-based systems. Moreover, the interleaving semantics is relaxed in the sense that the imported units may now operate on local components of actual configurations rather than on the global objects. The paper [KK00c] proposes an approach to design modular systems with heterogeneous components i.e. the transformation units and the transformation modules of a system may be based on different semantic domains and different types of rules and rule applications as long as the formats of the configurations are compatible with each other. Transformation units are the basic components of the graph-transformation-based specification language GRACE which is being designed to support the modeling of a wide spectrum of graph and graphical processes in a structured and uniform way. In [KBK00], some basic ideas and features of the graph-transformation-based specification language GRACE are sketched.

An important issue in the area of software architecture is the specification of reconfiguration and mobility of systems. Paper [HIM00] presents an approach for the specification of software architecture styles using hyperedge replacement systems and for their dynamic reconfiguration using constraint solving. A system architecture is represented as a graph where edges are components and nodes are ports of communication. Then, a style is represented as a graph grammar where the instances of the style are the graphs generated by the corresponding grammar. The construction and dynamic evolution of the style are represented as context-free productions and graph rewriting. Reconfigurations are modeled by allowing the declaration, creation and matching of new nodes (i.e. ports of communication) and synchronization is achieved using constraint solving over the productions of the style grammar. In this way complex evolutions can be specified in a more expressive and compact form than using π-calculus style languages for mobility. A related approach is presented in [HM01], where synchronization is modeled directly in the graph transformation formalism, by using *Synchronized Hyperedge Replacement Systems*. The capability of creation and sharing of ports together with multiple simultaneous synchronizations provides a very powerful tool to specify more complex evolutions, reconfiguring multiple components by identifying specific ports.

4.1.2 Parallelism and Concurrency

In the thesis [Bal00] (already described in the previous report), some basic approaches to the concurrent semantics of Petri nets, are extended to graph transformation systems. More specifically, focusing on the so-called *double pushout* (DPO) *algebraic approach* to graph rewriting, the thesis provides graph transformation systems with truly concurrent semantics based on (concatenable) processes and on a Winskel's style unfolding construction, as well as with more abstract semantics based on event structures and domains.

A basic role is played by *contextual nets* (also called nets with read, activator or test arcs) and *inhibitor nets* (or nets with inhibitor arcs), two generalizations of Petri nets, already known in the literature, which reveal a close relationship with graph transformation systems. The study of *contextual nets* is deepened in [BCM01], where extending Winskel's seminal work on safe nets, the truly concurrent semantics of contextual nets is given via a chain of coreflections leading from the category of contextual nets to the category of finitary coherent prime algebraic domains. The paper introduces also a notion of concatenable (deterministic) process of contextual nets, showing a coherence result between the process and unfolding semantics. This work is generalized to inhibitor nets in [BBCP00b]. Due to the non-monotonicity of the enabling in nets with inhibitor arcs, the causal structure of computations is far more complex, and a new, very general, notion of event structure, called *inhibitor event structure*, is needed to faithfully describe them. Again process semantics for inhibitor nets naturally induced by the developed theory is shown to agree with previous proposals in the literature, and to be consistent with the unfolding semantics. A wider treatment of the results can be found in the full version [BBCP00a].

According to the so-called "Petri nets are monoids" approach, the multiset structure of markings can be lifted to the computation level to characterize the concurrent net behaviors algebraically. In [BS00] (full version in [BS01]) the "Petri nets are monoids" approach is extended to contextual nets by considering monoidal categories of computations over a non-free monoid of places: a token is regarded as an *atom* that can emit several "negative" particles that can be read (but not consumed) separately.

4.1.3 Semantics, Correctness and Optimisation

The paper [Kle01], which was already described in the Third Annual Report, has been accepted for publication in *Mathematical Structures in Computer Science*. The paper [KV00] is described in the 2nd Annual Report.

It is common (e.g., in the meta model of UML) to use graphs for representing the abstract syntax of diagrams. Generalising term-based techniques for defining the semantics of (textual) programming languages we develop graph-based approaches to the semantics of diagram languages [EH00a, HE00]. In particular, we have generalised Plotkin's structural operational semantics (SOS) to the *graphical operational semantics* (GOS) approach [CHM00]. Denoting GOS rules as UML collaboration diagrams, the approach is used to specify the dynamic semantics of the UML [EHHS00, HS01, HS00, HZ01].

The paper [HP01] identifies a set of programming constructs ensuring that a programming language based on graph transformation is computationally complete. These constructs are (1) nondeterministic application of a set of graph transformation rules, (2) sequential composition and (3) iteration. This language is minimal in that omitting either sequential composition or iteration results in a computationally incomplete language. Computational completeness refers to the ability to compute every computable partial function on labelled graphs. The completeness proof is based on graph transformation programs which encode arbitrary graphs as strings, simulate Turing machines on these strings, and decode the resulting strings back into graphs.

4.1.4 Language Design

With PROGRES, two successful principles are combined in one formalism: rules and graphs (see [Mün00]). This is the common vision of the graph transformation community and the main motivation behind the development of PROGRES, a visual language that supports PROgramming with Graph REwriting Systems. In the past years, the PROGRES language has been considerably extended and enhanced. The main goal was to add appropriate support for programming and specifying in the large.

After having added a package concept (see last year's APPLIGRAPH report) the polymorphic typing concept has now been extended. PROGRES is a statically typed language. The integrated analysing tool is able to validate a PROGRES specification with more than 300 rules context-sensitively. With the help of a two level typing concept, a polymorphic typing system has been implemented. This typing system has been extended recently by the so-called co-variance of attribute type definitions regarding inheritance. The paper [MM00] explains these developments.

Based on this module concept and the research results of the evaluation of this concept, the development of a "true" OO extension of PROGRES has been started, i.e. graph transformations have been added to node class definitions as commonly known from other OO languages such as e.g. C++. This need has shown up pretty quickly after having studied an important application domain of PROGRES, visual programming and modelling languages. Currently, a student of Aachen University of Technology does his diploma thesis on this research topic.

In [GKT00], visual control concepts are discussed which are necessary to make the application of graph transformation more feasible for practical engineering and specification tasks. For this purpose control graphs based on the well-known activity diagrams are introduced. Moreover, the implementation of this control concept in AGG is presented showing the practical dimension of the concepts.

4.2 Tool Issues

The APPLIGRAPH subarea *Tool Issues* coordinates the activities of all partners concerning the design and realization of graph transformation tools. In the following, the recent tool building activities within APPLIGRAPH are listed.

4.2.1 DiaGen

For rule-based programming with graphs, graph transformation has to be extended by some further concepts. Concepts like abstraction, control, and encapsulation have been studied in [Hof00, HM00c].

The DiaGen tool developed at Universität Erlangen can generate editors that construct diagrams for a specific diagram language, and represent them by (hierarchical) graphs internally. With DiaGen, programming with graphs can be extended to *programming with diagrams*, using the visual notation of the program's application domain [HM00d, HM00a, HM00b]. For instance, Statecharts can be edited with an editor generated by DiaGen, and can then be animated or transformed by rule-based programming of their internal representations as hierarchical graphs [MH01].

4.2.2 GenGEd

In [NB00] GenGEd is presented. This system is used to interactively specify and generate syntax-directed editors for visual languages. In analogy to textual languages a visual language is specified by both an alphabet and a grammar. Hence, the GenGEd system provides an alphabet and a grammar editor. The grammar rules specify not only language generating productions but additionally the edit commands of the visual language editor. The generated editor can be used in various environments to allow for syntax-based drawing of diagrams.

4.2.3 PROGRES

Recent developments of PROGRES are a complete reimplementation of static type analyses of the PROGRES system to allow covariant inheritance on node attributes. That means that we are able to redefine attribute types. This development is a step towards generic modelling with PROGRES, i.e. implementing templates and graph classes (see [MS00] for details). The final step in that direction for modelling graph classes will be started soon: the implementation of true object-orientation. Node classes will be enriched by methods. Discussions about the design and usefulness have been very positive and publications on this topic are planned.

4.2.4 Graph Exchange Formats

The development of common exchange formats for graphs and graph transformation systems is an ongoing initiative within the EU Working Group APPLIGRAPH. The paper [Tae01] reports on the current state of this format discussion. The formats are based on the extensible markup language XML developed to interchange documents of arbitrary types. Graphs are basic structures in various areas of computer science. A common format for graphs supports the interaction of software developers building tools wrt. graph layout, graph algorithms, graph transformation, reengineering, etc. Moreover, within the graph transformation community tool builders can even gain from a common exchange format for graph transformation systems.

4.3 Application Domains

In this subarea, the intention is to demonstrate the usefulness of graph transformation by case studies in various fields like definition of visual languages, database models, concurrent and distributed systems, software process modelling, implementation of programming languages.

4.3.1 Visual Languages

Visual modeling techniques including UML as well as graph and net based techniques are of growing interest for software system specification and development. The GenGEd approach developed at the Technical University of Berlin allows already the generic description of visual modeling languages based on formal graph transformation and graphical constraint solving techniques and tools. In [BEE00], the GenGEd approach is reviewed and extended in order to allow the description of dynamic behavior and animation of systems. The basic idea is to define visual behavior and animation rules on top of the rules defining the corresponding visual modeling language and to allow a domain specific layout for the animation view of the system. A simple version of a traffic light system is used as running example, where the system view is given by a visual modeling language based on Petri nets, and the animation view shows directly the dynamic changes of the colors of traffic lights at a street crossing.

The use of simulation within software development aims at validating the system, as well as at recognizing the performance aspects. A simulation model includes the description of the behavior of the systems, but also intended animation strategies (to visualize the simulation) and statistics collecting/evaluating procedures. In [BER00], a modular approach for animating simulation models is presented. Both the system's behavior and its animation are modeled by graph grammars. The simulation of the system and its animation are supported by tools. We discuss the integration of the tools PLATUS, designed to construct the simulation mode ls, and GenGEd, designed for the development of visual languages, in order to obtain an environment in which animation modules can be obtained (from GenGEd) and used (in PLATUS). Moreover, due to the modular structure, an animation can be exchanged easily, and can be also reused by other simulation models.

The papers [BST00a, BST00b] propose an extension of layered graph grammars (LGGs), which have been introduced for the definition of visual languages (VLs). Offering new constructs like negative application conditions (NACs) allows for more concise VL definitions. A new layering condition and critical pair analysis are the prerequisites for a new parsing algorithm for visual languages which avoids the exponential behaviour of LGGs in many cases. The new type of graph grammar is applied to the definition of a simplified form of Statecharts.

On one hand visual techniques are important for specification, modelling and programming, in short visual computing, and on the other hand they can be used to support component technology, evolution and the software development process, in short visual development of software components. [ET00a] points out the central role of graphs and graph transformation for visual computing and visual development of software components. Moreover, it discusses the need for consistent integration of both areas and to propose graph transformation as a suitable technique for this purpose. The general idea of this position paper is not to present new technical results but to propose new lines of research how to combine and extend current projects and proposals in the areas of visual computing and visual development of software respectively

Collage grammars are an important means to specify visual languages. While a 2-dimensional grid picture grammar may generate pictures (defined as subsets of the unit square) with arbitrarily small details, only a finite number of them can be made visible as raster images for any given raster. In [DKEK00, DEKK00, DEKK01], an algorithm based on bottom-up tree automata is presented which computes the set of all raster images of the pictures generated by a given grid picture grammar.

From the perspective of computer science, interlacing knotwork, which forms a significant part of celtic art, is a visual language following mathematically precise rules of construction. In [DK00a, DK00b], the syntactic generation of celtic knots with collage grammars is studied. Several syntactic regulation mechanisms are employed in order to ensure that only consistent designs are generated. Practical research for the papers summarised above was carried out with the help of the system TREEBAG, which is briefly presented in [DK01].

Visual languages are often used in software development for modeling databases, software components, and the like. A particular application area, that has emerged in the last decade, is that of hypermedia application design. A hypermedia application usually provides information about a specific domain, and a navigation structure (hyperweb) that allows to browse through it. Both the underlying information and its navigation structure are normally designed using conceptual models based on visual languages (see e.g. RMM by Isakowitz Stohr and Balasubramanian, OOHDM by Schwabe, Rossi and Barbosa). While these languages provide very advanced features for modeling static aspects, they lack primitives for modeling transformations on a hyperweb and on its underlying information base. In [BtH00], an extension to the RMM model is proposed, where graph transformation systems are used for specifying such transformations. Future extensions concern, among others, the use of hierarchical graphs, as suggested by the OOHDM model.

4.3.2 Database Models and Information Systems

The activities within a computer system can be viewed as a sequence of operations on objects. One of the primary purposes of security mechanisms is *access control* which consists of determining and enforcing which active entities, e.g. processes, can have access to which objects and in which access mode.

In [KMPP00] we have presented a formalization of the Role-Based Access Control models using graph transformations. The advantages of this formalism include the presentation of a formal framework which allows the specification and verification of consistency requirements, the discussion of some open problems, such as the revocation cascade of users membership when administrative roles are decentralized, and a comparison of several possible solutions. The comparison of different Access Control policies within the graph transformation formalism and the analysis of the system behavior in combining different Access Control policies (such as a discretionary AC and a mandatory AC) are the subject of [KMPP01].

4.3.3 Concurrent and Distributed Systems

In order to account for the specific aspects of reactive and concurrent systems, graph transitions have been introduced as a variant of double-pushout graph transformation. Classical graph transformation systems come equipped with an implicit frame condition which ensures that nothing changed during the transformation beyond what is explicitly specified. They are not suitable for modeling interaction with the environment because they assume complete control about the system and its behaviour. Graph transitions, instead, provide a looser interpretation of graph transformation rules. A graph transition preserves, deletes and adds at least what is speicifed in the rule, but it allows additional changes which are interpreted as effects of operations of the environment. A truly concurrent semantics provides an explicit representation of parallelism. In [HLEO00] important results for a theory of concurrency in graph transition systems are presented and shown that they are highly relevant for the specification of reactive systems.

Distributed algorithms and dynamic distributed data structures are basic aspects of distributed systems which usually are modelled separately by different specification techniques. With the help of the running example of the mutual access to file systems, in [TE00] it is shown how typical problems are not handled by separate modelling. [TE00] also discusses how these problems can be solved by integrated specification of distributed algorithms and dynamic data structures by graph transformation. From the component point of view it is even more adequate to use distributed graph transformation for the semantics of distributed system specification.

Mobile and distributed systems typically consists of several layers: Objects on the lowest layer are clustered by components on the second layer, which themselves are located at nodes of a computer network on the third layer. Orthogonal to the three layers, an instance level and a type or schema level are distinguished when modelling these systems. Accordingly, the changes a system experiences during its lifetime can be classified as the system's dynamic behaviour on the instance level and as the evolution of the system on the schema level. In [EH00b, HE01] graph transformation is applied as conceptual and formal framework for describing the structural and behavioural aspects of such systems.

4.3.4 Agent-based Systems

In [DHK01b, DHK00b, DHK01a], typed graph transformation systems are used as a rule-based modelling notation for agent-based systems. As requirement specification, global graph transformation rules determine the overall effect of the interaction among agents and objects while abstracting from the communication involved. On the design level rules describe local, autonomous operations by which agents may react to changes in their environment. Graph transformation rules are controlled by means of state diagrams which describe the temporal ordering of an agent's local operations.

The approach is formalised in [DH00]. In [DHK00a, DHK00c], a concept of roles is introduced in order to model interaction protocols as well as the evolution of agents' structure and behaviour.

Agents and agent systems are a new means for structuring software. Concerning this approach an agent is the main abstraction, an entity encapsulating an inner state, a goal, and a set of rules. A first approach towards describing agents and agent systems using rule-based graph transformation is [KK00a]. The idea of a graph transformation-based specification of changes in environments and of agent systems in particular gave rise to further investigations in the context of logistics management ([EKT00, EK00]).

The foundations of a semantics for Agent Systems, based on graph rewriting, are investigated by D. Janssens (Antwerp) and P. Knirsch (Bremen). It has been shown, at least on the basis of a case study, that several notions relevant to agents, such as reactivity, autonomy and pro-active behaviour, can be expressed by features that are natural on the level of a graph rewriting system: termination conditions, priorities etc. These preliminary results point to the possibility to develop a GTS-based high-level language for agents.

4.3.5 Computer-Supported Cooperative Work

Several contributions to the semantic foundation of Computer-Supported Cooperative Work have appeared whose formal basis is Tile Logic. Tiles are rewrite rules with side-effects, and they are well-suited for modeling coordination languages, since they can be composed both statically and dynamically via possibly complex synchronization and work-flow mechanisms. By analogy with rewriting logic, the tile model has purely logical presentation, where tiles are considered as suitable sequents subject to simple inference rules.

In comparison with SOS formats, tile logic offers a straightforward extension of observational equivalence for open systems, which is finer and more elegant than the one obtained by closure under all ground substitutions. This point is made clear in [BdFEMOM00a] (full version in [BdFEMOM00b]), where, by analogy with the theory of SOS formats, the meta-theory of tile logic is investigated defining several formats guaranteeing that tile bisimilarity is a congruence.

Since configurations and observations come equipped with the same basic algebraic operations, tile logic is suitable for modeling *open ended systems*, where dynamic reconfiguration is obtained by observing suitable context embeddings. In particular, the approach can be stretched to deal with structural axioms on configurations, still guaranteeing the important property that "bisimilarity is a congruence" (as opposed to most SOS formats that often fail to ensure that bisimilarity is a congruence in the presence of structural axioms, making it necessary either to resort to context closure, which may result in a congruence which is no longer a bisimulation, or to consider *dynamic bisimilarity*, where context closure is performed during the bisimulation game, recovering the coarsest congruence which is also a bisimulation).

Zero-safe nets are a class of simple tile models which extend Petri nets with the notion of *transaction*—atomic computation obtained via the synchronization mechanism—and can provide a compositional semantic framework for many CCS-like calculi. In [BM01] the approach has been extended to deal with *read arcs* (which allow for multiple access in reading to the same token). Papers [BM00a, BM00b] summarize the theoretical foundation and the applications of zero-safe nets.

4.3.6 Design and Implementation of Programming Languages

The research group in Nijmegen investigates the suitability of the functional language CONCURRENT CLEAN for the timely development and maintanance of quality software. The CLEAN system runs on small machines and generates programs that run relatively fast. One special feature of the language is that due to *uniqueness typing*, interfacing the pure functional language CLEAN with the imperative world can be done in a flexible and efficient way. The current state of this project is that most of the libraries that have been developed by our group can be compiled with the new system. Since this is a rather large project, it is also a good test case for the quality of the generated code. A number of optimisations have been included ([vG00]) to generate even better code.

The Clean IDE has also been rewritten, now using the latest release of the *Object I/O* library [AW00] that allows one to construct programs with Graphical User Interfaces. This library makes good use of the uniqueness type system of Clean to interface with the imperative world. One distinct feature of this library is that interactive objects can have *local state*.

Besides having a high-quality compiler and programming environment, the success of a programming language is highly dependent on the availability of supporting libraries. We have released a number of libraries to make Clean useful not only inside the academic environment: *ArgEnv, CLAS, Directory, htoclean, MersenneTwister, Object I/O, 2D Platform Games, TCP, WrapDebug.*

4.3.7 Specification of Software Engineering Models

In object-oriented software modeling using the Unified Modeling Language (UML) different aspects of a system are represented by different diagram types. Static structure is modeled by UML class diagrams and interaction between different model components is modeled by UML interaction diagrams, i.e. by sequence diagrams or collaboration diagrams. Hence the complete description of a model consists of several diagrams of different diagram types. Therefore consistency of diagrams and consistency between different diagram types is an important issue.

In [Tsi00], consistency analysis between class and sequence diagrams based on attributed typed graphs and their transformation is described. More precisely, class diagrams are translated into an attributed typed graph called class graph and the multiplicity adornments into application constraints called multiplicity constraints. Sequence diagrams are represented by a graph grammar called interaction graph grammar. The consistency checking comprises existence, visibility and multiplicity checking. For consistency analysis consistency checking techniques of the algebraic theory of graph grammars using algebraic specifications for the attribute components are used to develop an algorithm which allows us to analyse and check this kind of consistency.

Part of the success of the Unified Modeling Language (UML) as a specification language is due to its diagrammatic nature. Its meaning is expressed by its meta model, a combination of class diagrams and constraints written in the Object Constraint Language (OCL), a textual language of expressions. Recent efforts have tried to give a formal semantics to OCL in a classical way. In [BKPPT00], a graph-based semantics for OCL and a systematic translation of OCL constraints into expressions over graph rules is proposed. Besides providing a semantical formalization of OCL, this translation can be employed to check the consistency of UML model instances wrt. the constraints, using a general purpose graph transformation machine like AG G or PROGRES. The translation of OCL constraints into graph rules suggests a way to express the constraints in a more intuitive visual form.

The evolution of software has become a more and more critical point in the use and maintenance of large software system. Accordingly, the continuous change of the system's design gets into focus. In [PEB00] and [EBP01] a two-fold approach combining visual architecture description with graphical specification techniques is suggested. Rule-based modification (a generalization of graph transformations) at both levels can be used to describe the model evolution. Allowing transformations at both levels allows distinguishing evolution steps with local and a global range.

Existing software configuration management systems embody a wide variety of policies for how artifacts can evolve. New policies continue to be introduced. Without a clean separation of configuration management policies from configuration management mechanisms, it is difficult to understand the policies as well as difficult to reason about how they relate. In [PW00], a formal foundation for specifying configuration management policies by viewing the policies in terms of graph transformation systems. Parisi-Presicce and Wolf are not only able to precisely capture the semantics of individual policies, but also to describe formal properties of the relationship between policies the first time.

The importance of embedded systems for our daily life is rapidly increasing. More or less unnoticed they fulfill the task of controlling the "behavior" of technical systems ranging from drinks' dispensers to big industrial plants. Embedded real-time systems play a special role. One important field of application for embedded real-time systems is in the telecommunications industry. The complexity of these systems is rapidly increasing while at the same time the software part is becoming more and more important. In the E-CARES (Ericsson Communication ARchitecture for Embedded Systems) research cooperation between Ericsson Eurolab Deutschland GmbH (EED) and the Department of Computer Science III, Aachen University of Technology, the subject of study is Ericsson's Mobile-service Switching Center (MSC) called AXE10. The cooperation aims to develop methods, concepts, and tools to support the processes of understanding and restructuring complex legacy telecommunication systems. The paper [HMJ00] reports on the latest developments on this topic. PROGRES is heavily used for implementing a reengineering prototype for the software of Ericsson's MSC. (For details about PROGRES please see [Mün00]).

In [Cre00] another project is described which is using PROGRES. Cremer describes an approach for reverse engineering existing applications with the aim to migrate an existing monolithic application to a distributed environment. This is done in two steps: First a COBOL85 application has to be reverse engineered. The result of that step is a cleanly structured distributable application. Secondly, a forward engineering has to be done to restructure the source code of the application according to the new architecture. The new source code was migrated to COBOL9x. Because of using PROGRES and TXL an adaptation to other reengineering problems is possible.

Another project which works with PROGRES deals with author support and is a collaborative research together with Springer publisher, Heidelberg, Germany. In today's publication process authors transform the ideas they wish to transport to the reader into linear text. The publisher then invests substantial manual labor to add semantical markup and bibliographical data to typeset and structure the document. The structured document is then printed and distributed to the reader who has to infer the intentions of the author from the printed text. We suggest graph-based tools for modeling the intended semantical structure of a text during document construction. The author uses these tools to create the semantical model stored as a graph (cf. [Gat00, GM00]).

The AHEAD project group is another main user of our PROGRES system. The core research area is the development of products in disciplines such as mechanical, electrical, or software engineering. Sophisticated tools for supporting the management of development products, processes, and resources are needed. One specific aspect of management systems is their ability to handle erroneous software development steps. While a software process is being executed, many errors and problems occur which require to reconsider previously executed process steps. In order to handle feedback in a process-centered software engineering environment, several requirements need to be addressed (adaptability, human intervention, impact analysis, change propagation, restoration of the work context, and traceability). Feedback management in our project DYNAMITE (DYNAMIc Task nEts) meets these requirements. The further development of DYNAMITE is called AHEAD (Adaptable Human-centered Environment for the mAnagement of Development Processes). Both systems are implemented in PROGRES.

Existing software configuration management systems embody a wide variety of policies for how artifacts can evolve. New policies continue to be introduced. Without a clean separation of configuration management policies from configuration management mechanisms, it is difficult to understand the policies as well as difficult to reason about how they relate. In [PW00], a formal foundation for specifying configuration management policies by viewing the policies in terms of graph transformation systems. In such a framework, it is possible not only to precisely capture the semantics of individual policies, but also to describe formal properties of the relationship between policies the first time.

4.3.8 Term Graph Rewriting

Building on term graphs and other algebraic structures for expressing formalisms different from the standard tree-like presentation of terms, it has been possible to identify a kernel of ingredients for describing normal form presentations of such formalisms (namely, *links*, *interfaces* and *modules*), showing that they all form different flavors of enriched symmetric monoidal categories [BGM00].

Relations, multirelations, partial functions, equivalences can be seen as particular instances of the more general concept of *span* (over **Set**). A span consists of a support (a set) and two legs (two arrows in **Set** having the support as source). In [BG01] we investigate the gs-monoidal structures of several span categories, making the correspondence with other classical categories such as **Rel** precise.

4.3.9 Application to Neural Networks

In [BFK00], attributed graph transformations are used to model neural networks. The paper follows a series of previous papers modeling neural networks and showing their termination. This paper emphasizes the labeling of the network structure which is the key in modeling a neural network. A common problem in many graph transformation approaches is the relabeling of objects. An existing solution is for example the introduction of auxiliary graph structures. In [BFK00], a double pushout approach to attributed graph transformation is presented that allows an intuitive way of relabeling due to a partial attribution.

References

[AW00] Peter Achten and Martin Wierich. A Tutorial to the Clean Object I/O Library - Version 1.2. Technical Report CSI-R0003, Computing Science Institute, Faculty of Mathematics and Informatics, University of Nijmegen, February 2000.

[Bal00] P. Baldan. *Modelling concurrent computations: from contextual Petri nets to graph grammars.* PhD thesis, Department of Computer Science, University of Pisa, 2000. Available as technical report n. TD-1/00.

[BBCP00a] P. Baldan, N. Busi, A. Corradini, and G.M. Pinna. Domain and event structure semantics for Petri nets with read and inhibitor arcs. Technical Report TR-05-00, Department of Computer Science, University of Pisa, 2000.

[BBCP00b] P. Baldan, N. Busi, A. Corradini, and G.M. Pinna. Functorial concurrent semantics for Petri nets with read and inhibitor arcs. In C. Palamidessi, editor, *CONCUR'00 Conference Proceedings*, volume 1877 of *LNCS*, pages 442–457. Springer Verlag, 2000.

[BCM01] P. Baldan, A. Corradini, and U. Montanari. Contextual Petri nets, asymmetric event structures and processes. To appear in *Information and Computation*. (Also Technical Report TR-99-18 of the Department of Computer Science, University of Pisa), 2001.

[BdFEMOM00a] R. Bruni, D. de Frutos-Escrig, N. Martí-Oliet, and U. Montanari. Bisimilarity congruences for open terms and term graphs via tile logic. In C. Palamidessi, editor, *Proceedings of CONCUR 2000, 11th International Conference on Concurrency Theory*, volume 1877 of *LNCS*, pages 259–274. Springer Verlag, 2000.

[BdFEMOM00b] R. Bruni, D. de Frutos-Escrig, N. Martí-Oliet, and U. Montanari. Tile bisimilarity congruences for open terms and term graphs. Technical Report TR-00-06, Computer Science Department, University of Pisa, 2000.

[BEE00] R. Bardohl, H. Ehrig, and C. Ermel. Generic Description, Behaviour and Animation of Visual Modeling Languages. In *Proc. Integrated Design and Process Technology (IDPT 2000)*, Dallas (Texas), USA, June 2000.

[BER00] R. Bardohl, C. Ermel, and L. Ribeiro. A Modular Approch to Animation of Simulation Models. In *Proc. 14^{th} Brazilian Symposium on Software Engineering*, Joao Pessoa, Brazil, October 2000.

[BFK00] M. Berthold, I. Fischer, and M. Koch. Attributed graph transformation with partial attribution and its application to neural networks. In *Proc. Joint APPLIGRAPGH/GETGRATS Workshop on Graph Transformation* [ET00b], pages 171–179.

[BG01] R. Bruni and F. Gadducci. Some algebraic laws for spans (and their connections with multirelations). In Gunther Schmidt Wolfram Kahl, David L. Parnas, editor, *Proceedings of RelMiS 2001, Relational Methods in Software*, pages 54–71, 2001. Bericht Nr. 2001-02, Fakultät für Informatik, Universität der Bundeswehr München. To appear in *ENTCS*.

[BGM00] R. Bruni, F. Gadducci, and U. Montanari. Normal forms for algebras of connections. *Theoretical Computer Science*, 2000. To appear.

[BKPPT00] P. Bottoni, M. Koch, F. Parisi-Presicce, and G. Taentzer. Consistency checking and visualization of ocl constraints. In A. Evans and S. Kent, editors, *UML 2000 - The Unified Modeling Language*, volume 1939 of *LNCS*. Springer, 2000.

[BM00a] R. Bruni and U. Montanari. First order and higher order tile models for open and mobile systems (abstract). In *Proceedings of TOSCA'00, Workshop Annuale del Progetto TOSCA*, 2000. Virtual Proceedings.

[BM00b] R. Bruni and U. Montanari. Zero-safe nets: Modeling transactions via transition synchronization (abstract). Technical Report CS-2000-01, Computer Science Department, University Ca' Foscari of Venice, 2000.

[BM01] R. Bruni and U. Montanari. Transactions and zero-safe nets, 2001. To appear in *Advances in Petri nets*.

[BS00] R. Bruni and V. Sassone. Algebraic models for contextual nets. In U. Montanari, J.D.P. Rolim, and E. Welzl, editors, *Proceedings of ICALP2000, 27th Int. Coll. on Automata, Languages and Programming*, volume 1853 of *LNCS*, pages 175–186. Springer Verlag, 2000.

[BS01] R. Bruni and V. Sassone. Two algebraic process semantics for contextual nets, 2001. To appear in *Advances in Petri Nets*.

[BST00a] P. Bottoni, A. Schürr, and G. Taentzer. Efficient Parsing of Visual Languages based on Critical Pair Analysis and Contextual Layered Graph Transformation. In *Proc. IEEE*

Symposium on Visual Languages, September 2000. Long version available as technical report SI-2000-06, University of Rome.

[BST00b] P. Bottoni, A. Schürr, and G. Taentzer. Efficient Parsing of Visual Languages based on Critical Pair Analysis and Contextual Layered Graph Transformation. Si-2000-06, University of Rome, 2000.

[BtH00] Giorgio Busatto and Pieter Jan 't Hoen. A graph-grammar based approach for the specification of hypermedia application dynamics. In Corradini and Heckel [CH00], pages 403–409.

[CH00] A. Corradini and R. Heckel, editors. *Proc. ICALP 2000 Workshop on Graph Transformation and Visual Modeling Techniques*, Geneva, Switzerland, July 2000. Carleton Scientific.

[CHM00] A. Corradini, R. Heckel, and U. Montanari. Graphical operational semantics. In Corradini and Heckel [CH00].

[Cre00] Katja Cremer. Graph-based reverse engineering and reengineering tools. In Manfred Nagl, Andy Schürr, and Manfred Münch, editors, *Application of Graph Transformations with Industrial Relevance*, LNCS 1779, pages 95–110. Springer-Verlag, 2000.

[DEKK00] Frank Drewes, Sigrid Ewert, Renate Klempien-Hinrichs, and Hans-Jörg Kreowski. Computing raster images from grid picture grammars. In M. Daley, M.G. Eramian, and S. Yu, editors, *Preproc. 5th Intl. Conference on Implementation and Application of Automata (CIAA 2000)*, pages 79–85. University of Western Ontario, London, Canada, 2000.

[DEKK01] Frank Drewes, Sigrid Ewert, Renate Klempien-Hinrichs, and Hans-Jörg Kreowski. Computing raster images from grid picture grammars. In M. Daley, M.G. Eramian, and S. Yu, editors, *Proc. 5th Intl. Conference on Implementation and Application of Automata (CIAA 2000)*, 2001. To appear.

[DH00] R. Depke and R. Heckel. Formalizing the development of agent-based systems using graph processes. In Corradini and Heckel [CH00], pages 419–426.

[DHK00a] R. Depke, R. Heckel, and J. M. Küster. Integrating visual modeling of agent-based and object-oriented systems. In *Proc. Fourth Intern. Conference on Autonomous Agents (AGENTS-2000)*, pages 82–83, 2000.

[DHK00b] R. Depke, R. Heckel, and J. M. Küster. Modeling agent-based systems with graph transformation. Technical Report No. 213, University of Paderborn, Dept. of Comp. Sci., August 2000.

[DHK00c] R. Depke, R. Heckel, and J.M. Küster. Requirement specification and design of agent-based systems with graph transformation, roles and UML. In P. Ciancarini and M.J. Wooldridge, editors, *Proc. 1st Int. Workshop on Agent-Oriented Software Engineering (ICSE-2000 Workshop)*, Limerick, Ireland, June 2000. University of Limerick.

[DHK01a] R. Depke, R. Heckel, and J. M. Küster. Improving the agent-oriented modeling process with roles. In *Proc. Fifth Int. Conference on Autonomous Agents (AGENTS-2001)*, Montreal, Canada, June 2001. To appear.

[DHK01b] R. Depke, R. Heckel, and J.M. Küster. Agent-oriented modeling with graph transformation. In P. Ciancarini and M.J. Wooldridge, editors, *Proc. 1st Int. Workshop on Agent-Oriented Software Engineering (AOSE 2000), Limerick, Ireland, June 2000*, volume 1957 of *LNCS*, pages 105–120. Springer-Verlag, Berlin, 2001.

[DK00a] Frank Drewes and Renate Klempien-Hinrichs. Picking knots from trees. the syntactic structure of celtic knotwork. Technical Report 4/00, Universität Bremen, 2000.

[DK00b] Frank Drewes and Renate Klempien-Hinrichs. Picking knots from trees. The syntactic structure of celtic knotwork. In M. Anderson, P. Cheng, and V. Haarslev, editors, *Proc. 1st Intl. Conference on Theory and Application* *of Diagrams 2000*, volume 1889 of *Lecture Notes in Artificial Intelligence*, pages 89–104. Springer, 2000.

[DK01] Frank Drewes and Renate Klempien-Hinrichs. Treebag. In M. Daley, M.G. Eramian, and S. Yu, editors, *Proc. 5th Intl. Conference on Implementation and Application of Automata (CIAA 2000)*, 2001. To appear.

[DKEK00] Frank Drewes, Renate Klempien-Hinrichs, Sigrid Ewert, and Hans-Jörg Kreowski. Computing raster images from grid picture grammars. Technical Report 2/00, Fachbereich Mathematik/Informatik, Universität Bremen, 2000.

[DKKK00] Frank Drewes, Peter Knirsch, Hans-Jörg Kreowski, and Sabine Kuske. Graph transformation modules and their composition. In M. Nagl and A. Schürr, editors, *Proc. Applications of Graph Transformation with Industrial Relevance (AGTIVE'99)*, volume 1779 of *Lecture Notes in Computer Science*, pages 15–30, 2000.

[EBP01] C. Ermel, R. Bardohl, and J. Padberg. Visual Design of Software Architecture and Evolution based on Graph Transformation. In *Proc. Uniform Approaches to Graphical Process Specification Techniques (UNIGRA'01)*, Genova, Italy, April 2001.

[EH00a] G. Engels and R. Heckel. From trees to graphs: Defining the semantics of diagram languages with graph transformation. In Corradini and Heckel [CH00].

[EH00b] G. Engels and R. Heckel. Graph transformation as unifying formal framework for system modeling and model evolution. In *Proc. ICALP 2000*, Lecture Notes in Computer Science, Geneva, Switzerland, 2000. Springer-Verlag.

[EHHS00] G. Engels, J.H. Hausmann, R. Heckel, and St. Sauer. Dynamic meta modeling: A graphical approach to the operational semantics of behavioral diagrams in UML. In A. Evans, S. Kent, and B. Selic, editors, *Proc. UML 2000, York, UK*, volume 1939 of *Lecture Notes in Computer Science*, pages 323–337. Springer-Verlag, 2000.

[EK00] Jens Eschenbächer and Peter Knirsch. Reengineering of supply chain networks with change management. In *Global*

Logistics for the New Millennium – Proc. of the 5th Internat. Symposium on Logistics (ISL 2000), Iwate, Japan, pages 198–207, 2000.

[EKT00] Jens Eschenbächer, Peter Knirsch, and Ingo J. Timm. Demand chain optimisation by using agent technology. In *IFIP WG 5.7 – International Conference on Integrated Production Management: Information and Communication Technology in Logistics and Production Management, Tromsø, Norway*, pages 285–292, 2000.

[ET00a] H. Ehrig and G. Taentzer. A Proposal for Consistent Integration of Visual Computing with Visual Development of Software Components. In *Proc. of Satellite Workshops of 27th Int. Colloqium on Automata, Languages, and Programming (ICALP'2000)*, pages 427–434, Geneva, Switzerland, July 2000. Carleton Scientific, Canada.

[ET00b] Hartmut Ehrig and Gabi Taentzer. Proc. Joint APPLIGRAPGH/GETGRATS Workshop on Graph Transformation. Technical Report 2000-2, Computer Science Department, Technical University of Berlin, March 2000.

[Gat00] Felix Gatzemeier. Patterns, schemata, and types—author support through formalized experience. In Bernhard Ganter and Guy W. Mineau, editors, *Proc. International Conference on Conceptual Structures 2000*, volume 1867 of *Lecture Notes in Artificial Intelligence*, pages 27–40. Springer-Verlag, 2000.

[GKT00] S. Gruner, M. Kurt, and G. Taentzer. A Visual Modeling Technique for Controlling Graph Transformations. In *Proc. of Satellite Workshops of 27th Int. Colloqium on Automata, Languages, and Programming (ICALP'2000)*, pages 435–442, Geneva, Switzerland, July 2000. Carleton Scientific, Canada.

[GM00] Felix Gatzemeier and Oliver Meyer. Making complex document structures accessible through templates. In Susan B. Jones and Beth Weise Moeller, editors, *Proc. 18th Annual Conference on Computer Documentation*, pages 509–519. IEEE Press, 2000.

[HE00] R. Heckel and G. Engels. Graph transformation and visual modeling techniques. *Bulletin of the European Association of Theoretical Computer Science (EATCS)*, 71:186–202, 2000.

[HE01] R. Heckel and G. Engels. Graph transformation as a meta language for dynamic modeling and model evolution. In T. Mens and M. Wermelinger, editors, *International Special Session on Formal Foundations of Software Evolution, Lisboa, Portugal*, March 2001. Co-located with the European Conference on Software Maintainance and Reengineering (CSMR 2001).

[HIM00] D. Hirsch, P. Inverardi, and U. Montanari. Reconfiguration of software architecture styles with name mobility. In A. Porto and G.-C. Roman, editors, *Coordination 2000*, volume 1906 of *LNCS*, pages 148–163, 2000.

[HLEO00] R. Heckel, M. Llabres, H. Ehrig, and F. Orejas. Concurrency of Double-Pullback Graph Transitions. In H. Ehrig and G. Taentzer, editors, *Proc. of Joint APPLIGRAPH/GETGRATS Workshop on Graph Transformation (GRATRA 2000)*, pages 146–154. Technical Report No. 2000-2, FB Informatik, TU Berlin, 2000.

[HM00a] Berthold Hoffmann and Mark Minas. A generic model for diagram syntax and semantics. In José D. P. Polim, Andrei Z. Broder, Andrea Corradini, Roberto Gorrieri, Reiko Heckel, Juraj Hromkovic, Ugo Vaccaro, and Joe B. Wells, editors, *ICALP Workshops 2000*, number 8 in Proceedings in Informatics, pages 443–450. Carleton Scientific, 2000.

[HM00b] Berthold Hoffmann and Mark Minas. Towards generic rule-based visual programming. In Margaret M. Burnett et al., editor, *Proceedings of the International Conference on Visual Languages (VL'2000)*. IEEE Press, 2000.

[HM00c] Berthold Hoffmann and Mark Minas. Towards rule-based visual programming of generic visual systems. In Nachum Dershowitz and Claude Kirchner, editors, *Proc. Workshop on Rule-Based Languages*, 2000. To appear.

[HM00d] Berthold Hoffmann and Mark Minas. Über generisches visuelles programmieren. In Stephan Diehl and Andreas Kerren, editors, *Proc. GI-Workshop Software Visualization*, Technical Report A/01/2000, pages 41–52. Universität des Saarlandes, 2000. In German.

[HM01] D. Hirsch and U. Montanari. A graphical calculus for name mobility. In *Proc. Workshop on Software Engineering and Mobility*, 2001. Co-located with ICSE 2001, Toronto. To appear.

[HMJ00] Dominikus Herzberg, Andre Marburger, and Tony Jokikyyny. E-CARES research project: Understanding complex legacy telecommunication systems. In Jürgen Ebert, Bernt Kullbach, and Franz Lehner, editors, *2nd Workshop on Software Reengineering*, Fachberichte der Universität Koblenz-Landau, 2000.

[Hof00] Berthold Hoffmann. From graph transformation to rule-based programming with diagrams. In Manfred Nagl, Andy Schürr, and Manfred Münch, editors, *Int'l Workshop on Applications of Graph Transformations with Industrial Relevance (*AGTIVE*'99), Selected Papers*, number 1779 in Lecture Notes in Computer Science, pages 165–180. Springer, 2000.

[HP01] Annegret Habel and Detlef Plump. Computational completeness of programming languages based on graph transformation. In *Proc. Foundations of Software Science and Computation Structures (FOSSACS 2001)*, volume 2030 of *Lecture Notes in Computer Science*, pages 230–245. Springer-Verlag, 2001.

[HS00] R. Heckel and S. Sauer. Dynamische Metamodellierung als Methode zur Definition einer operationalen Semantik für die UML. *Softwaretechnik-Trends*, 20(2):43–44, 2000.

[HS01] R. Heckel and St. Sauer. Strengthening UML collaboration diagrams by state transformations. In H. Hußmann, editor, *Proc. Fundamental Approaches to Software Engineering (FASE'2001), Genova, Italy*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[HZ01] R. Heckel and A. Zündorf. How to specify a graph transformation approach: A meta model for FUJABA. In H. Ehrig and J. Padberg, editors, *Uniform Approaches to Graphical Process Specification Techniques, satellite workshop of ETAPS 2001, Genova, Italy*, 2001.

[KBK00] Hans-Jörg Kreowski, Giorgio Busatto, and Sabine Kuske. GRACE as a unifying approach to graph-transformation-based specification. In *Proc. Uniform Approaches to Graphical Process Specification Techniques*, volume 44 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, 2000. To appear.

[KK00a] Peter Knirsch and Hans-Jörg Kreowski. A note on modeling agent systems by graph transformation. In M. Nagl and M. Schürr, A. amd Münch, editors, *Proc. Applications of Graph Transformations with Industrial Relevance (AGTIVE'99)*, volume 1779 of *Lecture Notes in Computer Science*, pages 79–86. Springer, 2000.

[KK00b] Hans-Jörg Kreowski and Sabine Kuske. Note on approach-independent structuring concepts for rule-based systems. In Hartmut Ehrig and Gabi Taentzer, editors, *Proc. Joint Appligraph and GETGRATS Workshop on Graph Transformation Systems*, Technical Report Nr. 2000-2, Technical University of Berlin, pages 41–49, 2000.

[KK00c] Hans-Jörg Kreowski and Sabine Kuske. Suggestions on the modularization of rule-based systems. In Martin Wirsing, Martin Gogolla, Hans-Jörg Kreowski, Tobias Nipkow, and Wolfgang Reif, editors, *Proc. Rigorose Entwicklung software-intensiver Systeme*, Technical Report Nr. 0005, University of Munich, pages 73–82, 2000.

[Kle01] Renate Klempien-Hinrichs. Normal forms for context-free node-rewriting hypergraph grammars. *Mathematical Structures in Computer Science*, 2001. To appear in the special issue *GRATRA 2000*.

[KMPP00] M. Koch, L.V. Mancini, and F. Parisi-Presicce. A Formal Model for Role-Based Access Control using Graph Transformation. In F.Cuppens, Y.Deswarte, D.Gollmann, and M.Waidner, editors, *Proc. of the 6th European Symposium on Research in Computer Security (ESORICS 2000)*, number 1895 in Lect. Notes in Comp. Sci., pages 122–139. Springer, 2000.

[KMPP01] M. Koch, L.V. Mancini, and F. Parisi-Presicce. Foundations for a graph-based approach to the Specification of Access Control Policies. In F. Honsell and M. Miculan, editors, *Proc. Foundations of Software Science and Computation Structures (FoSSaCS 2001)*, volume 2030 of *Lect. Notes in Comp. Sci.* Springer-Verlag, 2001.

[KV00] Hans-Jörg Kreowski and Gabriel Valiente. Redundancy and subsumption in high-level replacement systems. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Proc. Sixth Intl. Workshop on Theory and Application of Graph Transformations (TAGT'98)*, volume 1764 of *Lecture Notes in Computer Science*, pages 215–227. Springer, 2000.

[MH01] Mark Minas and Berthold Hoffmann. Specifying and implementing visual process modeling languages with DiaGen. In *International Workshop on Uniform Approaches to Graphical Process Specification Techniques (UNIGRA'01)*, 2001. To appear.

[MM00] Andy Schürr Manfred Münch. Generic and polymorphic graph transformations: Supporting the definition of visual languages. In *Proc. Workshop on Graph Transformation and Visual Modeling Techniques*, pages 459–466. Carleton Scientific, 2000.

[MS00] Manfred Münch and Andy Schürr. Generic and polymorphic graph transformations: Supporting the definition of visual languages. In *Proc. Workshop on Graph Transformation and Visual Modeling Techniques*, Satellite Event of ICALP'2000, Genf, 15. Juli 2000, pages 459–466. Carleton Scientific, 2000.

[Mün00] Manfred Münch. PROgrammed Graph REwriting System PROGRES. In Manfred Nagl, Andy Schürr, and Manfred Münch, editors, *Application of Graph Transforma-*

tions with Industrial Relevance, LNCS 1779, pages 441–448. Springer-Verlag, 2000.

[NB00] M. Niemann and R. Bardohl. Tool-based Specification of Visual Languages and Graphic Editors. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000)*, number 1785 in LNCS, pages 456–470. Springer, 2000.

[PEB00] J. Padberg, C. Ermel, and R. Bardohl. Rule-Based and Visual Model Evolution using GenGEd. In *Proc. of Satellite Workshops of 27th Int. Colloqium on Automata, Languages, and Programming (ICALP'2000)*, pages 467 – 476, Geneva, Switzerland, July 2000. Carleton Scientific, Canada.

[PW00] F. Parisi-Presicce and A. Wolf. Foundation for software configuration management policies using graph transformation. In T.Maibaum, editor, *Foundations of Software Engineering (FASE 2000)*, number 1783 in LNCS, pages 304–318. Springer, 2000.

[Tae01] G. Taentzer. Towards Common Exchange Formats for Graphs and Graph Transformation Systems. In *Proc. of UniGra - Uniform Approaches to Graphical Process Specification Techniques, Satellite Workshop of ETAPS'01, Genova, Italy*, 2001. To appear in Electronic Notes of TCS.

[TE00] G. Taentzer and H. Ehrig. Semantics of distributed system specifications based on graph transformation. In *GI Workshop "Rigorose Entwicklung software-intensiver Systeme", Berlin*, 2000.

[Tsi00] A. Tsiolakis. Consistency Analysis of UML Class and Sequence Diagrams based on Attributed Typed Graphs and their Transformation. Technical Report 2000/3, Technische Universität Berlin, Department of Computer Science, 2000.

[vG00] John van Groningen. Optimizing recursive functions yielding multiple results in tuples in a lazy functional language. In P. Koopman and C. Clack, editors, *Proc. 11th International Workshop on the Implementation of Functional*

Languages (IFL'99), volume 1868 of *LNCS*, pages 59–76. Springer-Verlag, 2000.