Appligraph
Applications of Graph Transformation

ESPRIT Working Group No. 22565

Fifth Annual Progress Report

Hans-Jörg Kreowski and Detlef Plump
(editors)

July 2002

Contents

1 Introduction **2**

2 Summary of Activities **2**
2.1 Research 2
2.2 Meetings 3
2.3 Information Dissemination 3
2.4 Further Activities 4
2.5 Special Highlight 5

3 Scientific Results of the Last 13 Months **5**
3.1 Language Issues 5
3.1.1 Modularity 5
3.1.2 Parallellism and Concurrency 7
3.1.3 Semantics, Correctness and Optimization 8
3.1.4 Language Design 10
3.2 Tool Issues 12
3.2.1 GenGEd 12
3.2.2 The DiaGen Tool Set 13
3.2.3 The Fujaba Tool Set 13
3.2.4 The PROGRES Tool Set 14
3.2.5 Further Tool Development Activities 15
3.3 Application Domains 15
3.3.1 Visual Languages 15
3.3.2 Database Models and Information Systems 17
3.3.3 Concurrent and Distributed Systems 18
3.3.4 Computer-Supported Cooperative Work 19
3.3.5 Specification of Software Engineering Models 21
3.3.6 Term Graph Rewriting 23
3.3.7 Application to Molecular Computing 23

5 References **23**

1 Introduction

This report summarizes the activities in the final 13 months of the ESPRIT Working Group APPLIGRAPH, covering the period from April 1, 2001, to April 30, 2002. 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 Summary of Activities

2.1 Research

During the final phase of APPLIGRAPH, research was carried out according to the Working Group Programme as specified in Annex I of the contract.

The results of this research are documented in 13 papers published or accepted for publication in journals and books, 62 papers published or accepted for publication in conference and workshop proceedings, 2 doctoral dissertations and 5 technical reports. See also Subsection 2.3 for special journal issues and proceedings of workshops in the final report period. The scientific results are described in Section 3 and references to publications are given at the end of this report.

2.2 Meetings

The international workshops UniGra 2001, GT-VMT 2001 and AGT 2002, the International Colloquium on Graph Transformation and DNA Computing, a Subgroup Meeting and a Team Leader Meeting were organized or co-organized during the final period of the Working Group.

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

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

AGT 2002 – Final APPLIGRAPH Workshop on Applied Graph Transformation.
This final event of the Working Group took place as an open satellite workshop of the ETAPS 2002 Joint Conference in Grenoble (France) from April 6 to April 14, 2002.

International Colloquium on Graph Transformation and DNA Computing.
This colloquium took place on February 14, 2002, at the Technical University of Berlin to celebrate the award of an honorary doctorate to Prof. Grzegorz Rozenberg (Leiden University).

Subgroup Meeting on XML-based Graph Exchange Formats.
This meeting took place on March 7-8, 2002, at the University of Erlangen (Germany).

Sixth Team Leader Meeting.
This meeting 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).

2.3 Information Dissemination

In the final period 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) appeared as special journal issues *Mathematical Structures in Computer Science* 12:4 (Cambridge University Press, 2002) and *Science of Computer Programming* 44:2 (Elsevier Science, 2002).

Proceedings of UniGra 2001. The proceedings of the Workshop on Uniform Approaches to Graphical Process Specification Techniques (UniGra 2001) appeared as Volume 44, Issue 4 of *Electronic Notes in Theoretical Computer Science* (Elsevier Science, 2001).

Proceedings of GT-VMT 2001. The proceedings of the Second International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2001) appeared as Volume 50, Issue 3 of *Electronic Notes in Theoretical Computer Science* (Elsevier Science, 2002).

Proceedings of AGT 2002. The proceedings of the Workshop on Applied Graph Transformation (AGT 2002) were published by the University of Grenoble in a series of workshop proceedings devoted to the satellite events of the ETAPS 2002 Joint Conference.

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).

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`.

2.4 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. The First International Conference on Graph Transformation will take place in Barcelona from October 7–12, 2002.

2.5 Special Highlight

In February 2002 Grzegorz Rozenberg, one of the APPLIGRAPH team leaders, has been awarded an honorary doctorate by the Technical University of Berlin for his scientific achievements including his work on graph transformation.

3 Scientific Results of the Last 13 Months

The following description of the scientific achievements obtained in the last 13 months is structured according to three research areas of the Working Group Programme: *Language Issues*, *Tool Issues*, and *Application Domains*.

3.1 Language Issues

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

3.1.1 Modularity

In the last year, work on modular as well as hierarchical graph transformations has been continued. In particular, the following results have been obtained and published.

In [Kre02], the basic features of graph transformation are introduced as a kind of sight-seeing tour of the computational landscape which is based on the application of rules to graphs. In particular, constructing a functional relation on graphs by graph transformation and a novel type of parallel graph transformation are studied.

In [BKK01] a new hierarchical graph model is introduced which allows to structure large graphs into small components. The model is equipped with a rule-based transformation concept in order to describe dynamic system behavior.

In [KBK+02] the structuring concepts of the graph- and rule-centered language GRACE are sketched and illustrated, including transformation units, formal parameter units, transformation modules, and their composition to modular systems. In [Kus01] the structuring concept of transformation units is employed for the formalization of the dynamic semantics of UML state machines. [Kus02] extends transformation units by formal parameters. The resulting parameterized transformation units are used in [KKK02] for a first

case study which models the logistic Pickup-and-Delivery Problem based on graph transformation.

In [BH01], a notion of hierarchical graph transformation is used that is generic, i.e. not committed to a particular kind of graphs or graph transformation, and decouples the package structure from the underlying graph. The existing approaches of H-graph grammars and hierachical hypergraph transformation are then compared by translating them to the generic notion.

In [Hof01], extensions of graphs and graph transformation that are important for programming with graphs are studied: On the one hand, graphs are extended by a notion of hierarchy that supports value composition, and hierarchical graph transformation is defined in an intuitive way that resembles term rewriting. On the other hand, admissible shapes for hierarchical graphs are required to be specified by context-free graph grammars, in order to set up a type discipline for shapely hierarchical graph transformation.

In [HM01], a new computational model for rule-based programming with graphs and diagrams is described. Using existing nesting concepts for graphs, this model defines an intuitive way of nested graph transformation that is based on variable matching. Shape rules are introduced for specifying structural consistency conditions on nested graphs. Shape rules set up a decidable type discipline for a refined model of shapely nested graph transformation. Since the refined model is compatible with the diagram editor DiaGen, it can be extended by costomizable diagram interfaces so that it specifies rule-based diagram transformation.

In [DHM02], it is shown that the construction of shapely nested graph transformation steps can be reduced to solving the subgraph isomorphism and variable matching problem for the components of a structured graph, and reestrictions of the transformation rules are devised that improve efficiency.

It has been shown that the framework of Cat-modules, developed by M. Simeoni in her Ph.D. thesis, can be instantiated into a module concept for Local Actions Systems. In [Ver02] the mathematical constructions behind this result are elaborated; in particular, an equational proof is given for the correctness of the construction enabling one to replace a production by a refinement. It is demonstrated how the same result can be used to construct a compositional semantics for interactive systems by introducing a generalized notion of a production where the left-hand side is a process instead of a graph.

In [EO01] a general version of a formal component concept based on integrated model specifications has been presented. This component concept is generic with respect to several parameters, which can be instantiated according to a large variety of integrated data type and process modeling techniques. The component concept provides explicit import and export inter-

faces. Different notions of composition of components are defined via match of corresponding interfaces. The main results show consistency and compositionality for the semantics of components and their composition. This generalizes corresponding concepts and results for the algebraic module concept and module concepts for graph transformation systems to components for integrated data type and process modeling techniques. These concepts are further developed towards a transformation-based component framework in [EOB+02a, EOB+02b] and especially useful to be instantiated by graph- and net-based techniques. The main concepts are a self-contained semantics and internal correctness of components, based on a new idea of high-level constraints. Two main results concerning compositionality show that semantics and correctness for a system can be inferred from that of its components. The concepts are illustrated by a running example on modeling Java threads by high-level nets.

Update: [KBK01] and [DHP02] have now appeared, the latter paper was described in the 3rd Progress Report.

3.1.2 Parallellism and Concurrency

In [JV01] and [Ver02] it is shown that the process theory of Local Action Systems can be instantiated for the special case of Petri Nets. Considering various straightforward encodings of markings by discrete graphs, one obtains in a uniform way several versions of the theory. A number of these correspond to variants of Petri Nets studied in the literature. Thus one may obtains a unified view of processes for these variants, including Place/Transition nets, Elementary Net Systems, Coloured nets and Nets with Inhibitor Arcs.

Following TER nets, an approach to the modelling of time in high-level Petri nets, [GH02] proposes a model of time within (attributed) graph transformation systems where logical clocks are represented as distinguished node attributes. Corresponding axioms for the time model in TER nets are generalised to graph transformation systems and semantic variations are discussed. They are summarised by a general theorem ensuring the consistency of temporal order and casual dependencies.

The resulting notions of *typed graph transformation with time* specialise the algebraic double-pushout (DPO) approach to typed graph transformation. In particular, the concurrency theory of the DPO approach can be used in the transfer of the basic theory of TER nets.

3.1.3 Semantics, Correctness and Optimization

The semantical theory of (DPO) graph transformation systems (GTSs) has been extensively studied in the first phase of the project. By now it comprises a rich theory of concurrency, which includes the generalisation, to this more general setting, of several fundamental approaches proper of Petri net theory (e.g., process and unfolding semantics). Most of the semantics proposed for GTSs define a (possibly concurrent) operational model of computation, which gives a concrete description of the behaviour of the system in terms of non-effective (e.g., infinite, non-decidable) structures. Thus they cannot be used directly to reason about the modelled system. Indeed, these operational models are intended to represent the basis for the definition of more abstract semantics, which take into account only some aspects of interest for the system at hand, disregarding inessential details. At this level one can define effective techniques for checking the equivalence of systems with respect to the selected observations, for verifying if a system satisfies a given property or for synthesising a system satisfying a given property.

Following this line of research, the paper [BCK01] proposes a verification technique for GTSs based on the costruction of an approximation of the unfolding semantics. As an application, it is shown how the proposed technique permits to verify the absence of deadlocks in an infinite-state Dining Philosophers system. In [BCK02] such technique is used to prove a security property for a distributed system with mobility modelled as a graph transformation system.

The traditional double-pushout approach to graph transformation does not allow to change node labels in an arbitrary context. The paper [HP02b] proposes a simple solution to this problem, namely to use rules with partially labelled interface graphs and to match rules injectively. In [HMP01] it was shown that injective matching makes the double-pushout approach more expressive, and here that approach is further generalised. Besides solving the relabelling problem, the proposed framework allows to write rules with partially labelled left-hand sides which are equivalent to (possibly infinite) sets of rules in the traditional setting. Unlike previous work on rules with partially labelled graphs, the approach does not need any labelling condition on matching morphisms, nor does it exclude node merging rules.

Generalising the work for contextual nets in [BCM00], the paper [BCM02] proposes an abstract semantics for graph transformation systems based on history preserving bisimulation (HP-bisimulation), a behavioural equivalence which, differently from ordinary bisimulation, is sensible to the concurrency properties of the systems. Resorting to HD-automata, a variation of ordinary automata suited to deal with history-dependent formalisms, HP-bisimulation

is shown to be decidable for finite-state graph grammars and an algorithm is proposed to decide such equivalence. The paper presents also the logical counterpart of HP-bisimulation, i.e., a logic in the style of Hennessy-Milner adequate for HP-bisimulation.

The context-free syntax of programming languages like PASCAL and MODULA is often defined by means of syntax diagrams in a quite intuitive way. The syntactically correct programs are obtained by reading words along paths in syntax diagrams in a certain way. In [DK01b], hyperedge replacement graph languages are interpreted as sets of syntax diagrams, and the string languages definable in this way are investigated.

The idea motivating the doctoral thesis [Kle01] is to combine the two most important context-free (hyper)graph grammar approaches, namely confluent node rewriting and hyperedge rewriting, into a framework which maintains the strengths particular to the original approaches. For this, confluent node rewriting in hypergraphs is established as the most powerful known context-free hypergraph grammar approach. As node-rewriting grammars are not naturally confluent, a decision algorithm is developed. Subsequently, a combination of node rewriting with the well-studied hyperedge-rewriting approach is studied. These methods to rewrite hypergraphs offer adequate formal frameworks for Petri net refinements. Finally, simulations of context-free hypergraph rewriting as well as Petri net refinement by pullback rewriting provide a link to category theory. The major language-theoretic results of [Kle01] are concisely presented in [Kle02a], and in [BJK02], the basic definitions and results of the pullback approach to graph rewriting are reviewed.

It is common (e.g., in the meta model of UML) to use graphs for representing the abstract syntax of diagrams. Generalizing term-based techniques for defining the semantics of (textual) programming languages we develop graph-based approaches to the semantics of diagram languages. In particular, we have generalized Plotkin's structural operational semantics (SOS) to the *graphical operational semantics* (GOS) approach. Denoting GOS rules as UML collaboration diagrams, the approach is used to specify the dynamic semantics of the UML [HS01, HZ01, HHS01].

Object-oriented modeling favors the modeling of object behavior from different viewpoints and at different levels of abstraction. This gives rise to consistency problems between overlapping or semantically related submodels. The absence of a formal semantics for the UML and the numerous ways of employing the language within the development process lead to a number of different consistency notions. Therefore, general meta-level techniques are required for specifying, analyzing, and communicating consistency constraints. In this paper, we discuss the issue of consistency of behavioral models in the

UML and present techniques for specifying and analyzing consistency. Using meta-model rules we transform elements of UML models into a semantic domain. Then, consistency constraints can by specified and validated using the language and the tools of the semantic domain. In [HK01, EHK01] this general methodology is introduced and exemplified by the problem of protocol statechart inheritance while [EKGH01] applies the approach to consistency issues of UML-RT.

Update: The papers [Kle02b] and [HMP01] have now appeared, the latter was described in the Third Progress Report.

3.1.4 Language Design

With PROGRES, two successful principles are combined in one formalism: rules and graphs. 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 have been extended and enhanced considerably. The main goal was to add appropriate support for programming and specifying in the large.

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 implemened. This typing system has been extended recently by parametric polymorphism. Templates of node classes can be specified which are instantiated later by special parameters, maintaining the ability of static type checking.

A "true" object-orientation specification methodology of PROGRES has been developed, i.e. graph transformations have been added to node class definitions as commonly known from other OO languages such as e.g. C++, rather than allowing the definition of attributes to node classes only. This need has shown up pretty quickly after having studied an important application domain of PROGRES, visual programming and modelling languages. The use of object-oriented programming principles became already obvious when the first rapid prototype for a visual programming language was specified.

The paper [HP02a] shows how to program in a core programming language based on graph transformation rules. The presented programs compute various functions and relations on graphs: the functions generating the transitive closure of a graph and the disjoint union of all subgraphs of a graph, the relation yielding all spanning trees of a graph, and functions testing for connectedness, acyclicity, and planarity. The language has a simple syntax

consisting of just three constructs: nondeterministic one-step application of a set of rules, sequential composition, and iteration. It also has a simple formal semantics, and was shown to be computationally complete and minimal.

A paper *On diagrammatic languages and their editors* has been presented at DIAGRAMS 2002 ([BC02b]). The paper deals with a metalevel approach to the definition of families of visual languages. Diagrammatic languages are characterized by the existence of identifiable elements in some spatial relations. By progressively specifying the types of relations and the the types of elements which can be involved in them, subfamilies can be derived up to the definition of concrete syntaxes. The approach lends itself to the systematic derivation of rules or rule schemata to construct correct sentences or to parse them. The structure of this rule is independent of the specific language or family of languages, so both graph transformations, high level replacement, multiset or Chomsky-like grammars can be used. Syntax directed editors and parsers can be automatically constructed from such rewriting systems.

In [DHK01a], typed graph transformation systems are used as a rule-based modeling 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.

In [DHK01b] a concept of roles is introduced in order to model interaction protocols as well as the evolution of agents' structure and behavior.

In object-oriented software development, requirements of different stakeholders are often manifested in use case models which complement the static domain model by dynamic and functional requirements. In the course of development, these requirements are analyzed and integrated to produce a consistent overall requirements specification. Iterations of the model may be triggered by conflicts between requirements of different parties. However, due to the diversity, incompleteness, and informal nature, in particular of functional and dynamic requirements, such conflicts are difficult to find. Formal approaches to requirements engineering, often based on logic, attack these problems, but require highly specialized experts to write and reason about such specifications.

In [HH01] the use of graph transformations is proposed to formalize the relation between different aspects, i.e., an integrated model combining UML use case, activity, and collaboration diagrams. The formalization allows to make precise the notions of conflict and dependency between functional requirements expressed by different use cases. Then, use case models can be

statically analyzed, and conflicts or dependencies detected by the analysis can be communicated to the modeler by annotating the model. An implementation of the static analysis within a graph transformation tool is presented in [HHT02].

3.2 Tool Issues

The APPLIGRAPH subarea *tool issues* mainly deals with the design and development of graph transformation tools. This research is influenced by both application issues concerning application domains of the realised tools, and language issues concerning the implementability of specification and programming languages.

In this section we describe the activities of all APPLIGRAPH members and their scientific correspondents related to the development of tools.

3.2.1 GenGEd

In [BEE+02], an overview is given about the current concepts of GenGEd, an environment for the visual definition of visual languages (VLs). From the visual definition, a VL specification is obtained that serves as a configuration of a VL-specific environment, i.e., the configuration is dependent on the parameters available in a VL specification: GenGEd allows for the visual specification of syntax-directed editing, parsing, and simulation as well. In addition to these features, we show how to define an animation view for a certain VL model. All GenGEd features are based on the formal concepts of algebraic graph transformation and graphical constraint solving. Hence, we have a well-defined theory which serves as a basis for proper extensions of GenGEd.

GenGEd supports mainly the generation of syntax-directed editors. However, syntax-directed editing is not always desired by the user. Therefore in [BST01], GenGEd is extended by parsing facilities which allow for free editing as well. The implementation goes back to graph parsing in Agg which can use critical pair analysis to gain more efficiency.

The GenGEd approach allows for the generic description of visual modeling languages such as different Petri net classes. In [EBE01], GenGEd has been extended in order to support the description and implementation of animation views for different Petri net classes. A generic, domain-specific animation of net behavior, however, is not yet supported by the Petri Net Kernel (PNK). The PNK is a tool infrastructure for editing, simulating and analyzing Petri nets of different net classes and for integration of other Petri net tools. It is part of the so-called *Petri Net Baukasten*. In our framework,

the animation view of a system modeled as a Petri net consists of a domain-specific layout and an animation according to the firing behavior of the Petri net type. The basic idea is to generate visual behavior and animation rules based on visual syntax rules defining the corresponding Petri net language. For this purpose we propose a view transformation from the classical Petri net layout to the animation layout. The well-known producer/consumer system modeled as an elementary Petri net serves as running example. The animation view of the model in the application domain of a kitchen shows directly the production and consumption of cakes. Finally, the implementation of animation concepts in the GenGEd environment is presented, including a discussion about the integration of GenGEd and the PNK via a transformation of corresponding XML-based interchange formats.

3.2.2 The DiaGen Tool Set

Diagrams are frequently used to represent or to communicate complicated situations. Depending on the problem domain, diagrams have a specific graphical representation, syntax, and semantics. Diagrams with common syntax and semantics make up a diagram class. Graphical editors are required in order to create or edit diagrams on a computer. Diagram editors are graphical editors that are tailored to a specific diagram class.

DiaGen (*Dia*gram Editor *Gen*erator) is a framework that supports the user who has to write diagram editors. Diagrams are internally modeled by hypergraphs; hypergraph grammars are therefore a suitable means for describing the syntax of a diagram class. *DiaGen* comes with a generator which is used to create an editor based on a formal specification that mainly describes diagram syntax by this hypergraph grammar and editor behavior by graph transformation.

The paper [MH01b] describes how a diagram language can be specified, based on graphs, graph grammars, and transformation rules, and how the diagram editor generator DiaGen generates a diagram editor from such a specification. DiaGen can be applied to practically every visual language, and to visual process modeling languages in particular. This is demonstrated with an editor and animator for statecharts.

3.2.3 The Fujaba Tool Set

In previous reports we described the development of the Fujaba tool set which was realised by our scientific correspondents Albert Zündorf at the University of Paderborn. Work on this project did not continue because of the leave of Dr. Zündorf. Here, we briefly present Fujaba, details can be

found in previous annual reports.

The Fujaba project aims to provide a CASE tool that combines UML and Graph Grammars to a well defined object-oriented specification language. The Fujaba environment provides code-generation facilities that allow the generation of Java code from class diagrams and story diagrams. This Java code does not rely on a heavy-weight library or database but only on some standard container classes, i.e. the Java Generic Library from Objectspace. Thus, the generated code is easily integrated with other system parts, e.g. a graphical user interface.

The development of object-oriented applications starts with the refinement of UML's use-cases via so-called story boards. Story boards have a semantics close to story diagrams. However, while story diagrams are executable formal specifications that allow a direct translation to an object-oriented programming language, story boards describe only possible scenarios. Thus, story boards are used to outline first ideas about the internal behaviour of the desired system. In the next step the story boards are analysed and serve as a basis for the formal specification of the behaviour of the desired system using story diagrams. In the final phase the Fujaba environment generates code from the story diagrams that is comined with a graphical user interface and the other system parts.

3.2.4 The PROGRES Tool Set

The PROGRES-system is an integrated visual programming environment to support the definition and implementation of graph rewriting rules. Our main tasks are the maintenance and development of the PROGRES language and improving the usability of the PROGRES environment. In the past years PROGRES has been used in many research projects, e.g. for process modelling and management, workflow support, specifying CIM environments, versioning control and configuration management tools etc. The functionality of these systems was specified using PROGRES and code was generated automatically by a built-in compiler. This code was used in connection with the visual rapid prototyping framework UPGRADE to realise a graph-based stand-alone application implementing the specified functionality.

We collaborate with a number of partners from industry such as Springer Verlag, Ericsson, etc. We maintain contacts to user communities in Berlin, Erlangen-Nürnberg, Paderborn (all Germany), Kingston (Canada), Carleton University Ottawa (Canada), University of York (UK), INRIA research institute (France) etc. We also take part in the joint European research project APPLIGRAPH (Applications of Graph Transformation) which is sponsored by the European Community.

Current research activities comprised the improvement of the PROGRES language and environment. Last year we have integrated parametric polymorphic typing concepts to the PROGRES language. With polymorphic typing concepts we are able to specify generic graph class templates. The core research of the PROGRES group regarding application of graph transformation dealt with the definition of generic modules for editing, analysing, and interpreting visual languages. We have defined a set of reusable modules for a visual standardised language which is heavily used in automation industry, the IEC-61131/3. These modules can also be reused for the definition of other component-based visual languages as far as possible. Also, we have added the opportunity to define a behaviour of graph elements to the already existing object-based specification methodology, resulting in an object-oriented specification approach as known from modern languages like e.g. Java. Currently, we are evaluating this new specification methodology. Ongoing work also includes the transition of PROGRES' user interface to a modern GTK-based GUI.

3.2.5 Further Tool Development Activities

Last year, the development of common exchange formats for graphs and graph transformation systems has been pursued further. In [Tae01] the current state of these research activities is reported.The developed formats are based on the extensible markup language XML. A common format for the description of graphs supports the interaction of software developers building tools with respect to graph layout, graph algorithms, graph transformation, reengineering, etc. Moreover, within the graph transformation community tool builders can benefit from a common exchange format for graph transformation systems.

3.3 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 and software process modelling.

3.3.1 Visual Languages

The visual environment GenGEd supports the visual definition of visual languages (VLs). Each VL is defined by an alphabet and a grammar. From a specific VL-definition, a VL-specification is generated which is the input

of a graphical editor allowing for syntax-directed editing of diagrams over the specified VL. GenGEd as well as each VL is based on the well-defined concepts of algebraic graph transformation and graphical constraint solving. The underlying formalism is hidden from the user, but it is essential for a formal presentation and manipulation of graphical structures. In [Bar02], the main concepts of GenGEd and the underlying tool are presented and illustrated by the definition of a simple kind of the well-known statechart language.

In [BEE01], the description of dynamic behavior and animation of systems is presented. 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. Two visual language models based on the visual language of Petri nets serve as running examples: a simple version of a traffic light system and the well-known specification of The Dining Philosophers. The system views are given by means of high-level Petri nets. The animation view of the traffic light system shows directly the dynamic changes of the colors of traffic lights at a street crossing, whereas the animation view of The Dining Philosophers visualizes the behavior of philosophers sitting around a table.

The Platus simulation environment allows for a modular description of simulation models, clearly separating aspects of behavior, statistics and animation. Each component of a model in Platus has its own animation interface, that describes when and which messages will be sent to a corresponding animation module. In [BR01], GenGEd is extended in order to define several animation modules which can be connected via animation interfaces to Platus modules, allowing for a visual animation of the application being simulated.

On the Symposium on Visual Languages and Formal Methods being part of the IEEE Symposia on Human-Centric Computing Languages and Environments (HCC'01) there was a special statechart modeling contest. In [BT01] and [BE01], a slightly simplified version of statecharts as they are supported by the CASE tool Rhapsody is considered. In [BT01], contextual layered graph transformation as introduced in [BST01] is used to define high-level parsing rules and the operational semantics for this kind of statecharts. In [BE01], GenGED is used for the visual specification of both, syntax-directed editing commands of a language-specific graphical editor as well as for the definition of parsing rules based on contextual layered graph transformation.

In a rule-based approach the computation steps of a system are specified by rules that completely define how the system's state may change. For open systems a more liberal approach is required, where the state changes are only partly specified, and—interactively—other components may con-

tribute further information on how the transformation is defined completely. In [BGRS01], a formal model for interactive rule–based specifications is introduced, where states are modelled as partial algebras and transformations are given by internal algebra rewritings and arbitrary external components. As an application we discuss how visual languages can be defined in this framework. Thereby the internal (logical) representations of visual expressions are transformed by rewriting rules, whereas their layouts are obtained interactively by external components like a constraint solver or a user working with a display and a mouse.

In [BKPT01], a visualization of OCL is proposed within the context of the UML meta model, so that OCL expressions are represented by extending collaboration diagrams. The description of properties of objects is based on collaborations, where classifier and association roles are used to describe navigation paths. Operations computing properties are described by interactions consisting of messages between classifier roles. The introduction of new graphical core elements is kept to a minimum. New notation mainly concerns the predefined operations in OCL and provides more convenient visual forms for the notation by interactions here. The proposed visualization is described in detail and is illustrated with examples taken from an industrial project under development. Main ideas for this kind of visualization are derived from a graph-transformation based approach. Moreover in [BKPT02], we discuss how an operational semantics for OCL can be based on graph transformation units which guide the application of graph-transformation rules.

Update: [DK00] of the third interim report has now appeared [DK01a].

3.3.2 Database Models and Information Systems

One of the primary purposes of the security mechanisms in a computer system is to determine and enforce which active entities, e.g processes, can have access to which objects and in which access mode. This activity is called *access control* and in [KMP00, KMP01a] we have presented the advantages of a formalization of the Role-Based Access Control models and of other Access Control policies using graph transformations. A policy framework consists of a type graph, a set of graph rules (partial graph morphisms) and two sets of graphical constraints describing what is forbidden and what is required in the state graphs derivable with the rules. In [KMP01b] we have addressed the problem of combining the graph based specifications of two access control policies (even if based on different underlying access control models) while in [KMP02] we have studied how to detect and resolve conflicts between two rules, between two constraints, and between a rule and a constraint, deriving from the integration of two policies.

3.3.3 Concurrent and Distributed Systems

Distributed object computing is a computing paradigm that allows objects to be distributed over a heterogeneous network. Infrastructures help to develop distributed object applications by offering necessary services for distributed computing. Having a comprehensive infrastructure to hand, the development of complex distributed object systems is feasible in principle. Flexibly evolving architectures as well as highly dynamic distributed object structures are key requirements for nowadays distributed solutions. They can hardly be well designed on this level of programming, due to their complexity. In [Tae02b], , the graphical description of the network and its dynamic reconfiguration as well as the component interfaces and local object systems and their behavior is shown. Typical issues in distributed systems like remote object interaction, object migration and replication, communication and synchronization are expressible in this technique. The notation is close to UML. It extends the UML notation where needed. In [Tae02a], a visual framework is presented which offers a more abstract and intuitive approach to the relevant aspects of a distributed object system. In this framework, the UML-like notation, its semantics based on graphs and their transformation, and more abstract on I/O-automata, are distinguished. The main aim of [BEH+02] is to give an overview concerning applications of graph transformation techniques to the area of Petri nets achieved by the team of TU Berlin within the APPLIGRAPH Working Group and the DFG Researcher Group on Petri Net Technology. In this overview, net transformation systems are considered as instatiation of high-level replacement systems, property preserving net transformations module and component concepts and the modelling of open systems.

In [JV01] and [Ver02] The relationship between Local Action Systems and Petri Nets is investigated. Apart from the aspects reported in section 1.2, the framework obtained by specializing Local Action Systems to the rewriting of discrete graphs suggests possible extensions of Petri nets, improving their expressive power. As an example, in [Ver02] an extension is considered where dynamic changes of the net can be taken into account. The same thesis also contains a number of examples where the decomposition techniques for Local Action Systems are used to provide formal correctness proof of distributed systems. The examples include a distributed shortest path algorithm and a simple transport protocol.

In [KM01] a notion of bisimulation for graph rewriting systems is introduced, allowing one to prove observational equivalence for dynamically evolving graphs and networks. This notion is applied to the framework of synchronized graph rewriting with mobility which is describe in two differ-

ent, but operationally equivalent ways: on graphs defined as syntactic judgements and by using tile logic. One of the main results of the paper says that bisimilarity for synchronized graph rewriting is a congruence whenever the rewriting rules satisfy the basic source property.

In [FMT01] a simple labelled transition system semantics of Cardelli and Gordon's Ambient calculus is presented, by exploiting a general and flexible model based on (hyper)graphs, where graph transformation is obtained via (hyper)edge replacement and local synchronization with mobility. In addition to tree-like ambients, proposed calculus works just as well with graph-like ambients, which are a more realistic model of internetworks.

3.3.4 Computer-Supported Cooperative Work

Strongly connected with the idea of a graph-transformation-based specification of changes in environments and of agent systems in particular, investigations in the context of logistics management have been furthered in the direction of Mass Customisation [TKH+01, TWK+01].

Tile Logic [GM00] is a rule-based framework that exploits a three-dimensional view of concurrent systems: the horizontal dimension (space) is devoted to the modeling of states and components; the vertical dimension (time) models computation steps; and the third dimension (concurrency) accounts for the distribution of activities and resources. This separation of concerns makes it possible to select different flavors of Tile Logic simply by fine-tuning the algebraic structure of the elements in each of the three dimensions and by fixing their interplay. In fact, several contributions to the semantic foundation of Tile Logic and its application to Computer-Supported Cooperative Work have been developed that allow to investigate different features and properties.

In [BMM02], two classes of Tile Logic are considered, called Process Tile Logic (PTL) and Term Tile Logic (TTL), where the horizontal structure is completely analogous to the vertical structure and consists of a special kind of acyclic graphs in the case of PTL and of ordinary terms in the case of TTL. Categorical models based on double categories are then defined for both PTL and TTL.

In [BM02], Tile Logic is exploited to combine different kind of graph *connectors* in space and time, accounting for graph-like representations of resource/location sharing and causal dependencies. The result is the formalization of a meta-theoretic approach to the definition of operational and abstract concurrent semantics for systems with name handling. A detailed case study shows that the straightforward application of the general framework to basic parallel processes yields a concurrent operational semantics whose

observations exactly characterize the ordinary causal weak bisimilarity based on Darondeau and Degano's causal trees.

In [BMR01], Tile Logic is applied to Logic Programming by exploiting a finitary presentation of unification-as-pullback to define the coordination rules between goals and concurrent refutations. The result is the view of Logic Programming as an interactive system, equipped with concurrent and coordinated abstract semantics consistent with the most common semantics of logic programming. Moreover, the compositionality of the tile semantics is guaranteed by standard results, as it reduces to check that the tile systems associated to logic programs enjoy the *tile decomposition property*. An extension of the approach for handling constraint systems is also discussed.

The paper [BHLM01] compares the higher-order version of Tile Logic with the Edinburgh Logical Frameworks, discussing the main features of the two approaches as *meta-languages* and *(computational) meta-models*, and in particular with their use as logic and semantic frameworks for the definition and analysis of languages, calculi and programming paradigms. The comparison traces differences and analogies between the two approaches on the basis of two case studies: late π-calculus and lazy simply typed λ-calculus.

Two contributions [BM01a, BM01b] regard a basic flavor of Tile Logic, called *zero-safe nets*, where both the horizontal and the vertical structure are just freely generated monoids. Zero-safe nets extend Petri nets with the notion of *transaction* and can provide a compositional semantic framework for many CCS-like calculi. In [BM01a], first zero-safe nets are extended with *read arcs* (which allow for multiple access in reading to the same token) and then a distributed interpreter is defined for the resulting framework by exploiting the net unfolding of ordinary Petri nets with read arcs.

The paper [BM01b] discusses the conditions under which *inhibitor arcs* can be introduced in zero-safe nets without compromising the abstract view of the system and applies *contextual zero-safe nets* to a transactional extension of the coordination language Linda for defining a concurrent operational semantics.

Another use of PROGRES in Aachen deals with author support (see [Gat01]). This direction aims at supporting authors to transform the ideas they wish to transport to the reader into linear text. Graph grammars are used for structuring documents and adding semantical markup and bibliographical data. Graph-based tools are used are used for modelling the semantic structure of a text when the text is being constructed already.

3.3.5 Specification of Software Engineering Models

In object-oriented software development, requirements of different stakeholders are often manifested in use case models which complement the static domain model by dynamic and functional requirements. In the course of development, these requirements are analyzed and integrated to produce a consistent overall requirements specification. Iterations of the model may be triggered by conflicts between requirements of different parties. However, due to the diversity, incompleteness, and informal nature, in particular of functional and dynamic requirements, such conflicts are difficult to find. Formal approaches to requirements engineering, often based on logic, attack these problems, but require highly specialized experts to write and reason about such specifications. In [HHT02], a formal interpretation of use case models is proposed consisting of UML use case, activity, and collaboration diagrams. The formalization, which is based on concepts from the theory of graph transformation, allows to make precise the notions of conflict and dependency between functional requirements expressed by different use cases. Then, use case models can be statically analyzed, and conflicts or dependencies detected by the analysis can be communicated to the modeler by annotating the model. An implementation of the static analysis within a graph transformation tool is presented.

The use of UML for software specification leads usually to lots of diagrams showing different aspects and components of the software system in several views. In order to support a view-oriented approach to system modeling, consistency in views and in between views has to be manageable. It is a reasonable approach to consistency management when first choosing a suitable semantic domain, provide a partial mapping into this domain, and specify as well as verify consistency constraints formulated in that domain. Annotated meta model rules can be used to translate elements of UML models into the semantic domain chosen. In [HKT02], triple graph grammars and attributed graph transformation approaches are considered for the precise definition of meta model rules. Moreover, the tool support for automatic translation is outlined.

In a software system development process a variety of heterogeneous viewpoint models of the system are constructed for the abstract specification of its functionality and behaviour. These have to be integrated in order to achieve a coherent and consistent global system specification. Transformation systems constitute a formal semantic domain where specification languages conforming to different paradigms can be interpreted, which makes possible a formal semantic integration of heterogeneous viewpoint models. In [GJS02], an overview of the transformation system approach is given. Anal-

ogously to graph transformation systems states are described by algebraic structures and behaviour by the ordering of state transformation steps, i.e., changes of the algebraic structures. In contrast with graph transformation systems, however, transformation systems are generic w.r.t. the choice of the state representations, i.e., instead of graphs or extensions thereof arbitrary structures and corresponding logics may be used. Moreover, development relations and structuring operations are defined, which altogether yields a powerful and flexible integration framework.

In [DJM02] the use of graph rewiting for the formalization of refactoring operations is explored. A graph representation of object-oriented programs is presented, and it is demonstrated that a number of refactoring operations considered in the literature can be modeled in an elegant way by graph productions. The examples are based on JAVA, but the main ideas are language-independent. This type of research is important because much of the work on refactoring is still at the semi-formal level, in spite of the fact that it is very important that refactorings guarantee a number of relevant program properties.

The importance of embedded systems for our daily life is constantly increasing. They fulfill the task of controlling the behaviour of technical systems ranging from small devices like e.g. PDAs, dispensers, etc. to big industrial plants. Embedded real-time systems play a special role. One of the most important fields of application for embedded real-time systems is to be found in the telecommunications industry. The ongoing research aims at developping methods, concepts, and tools to support the processes of understanding and restructuring complex legacy telecommunication systems (see also last year's report). Our graph rewriting system PROGRES is heavily used for implementing a reengineering prototype for the software of Ericsson's MSC. Publications of Marburger et al. (see [MH01a, MH02]) document the latest research results on this topic. The AHEAD project group is another main user of our PROGRES system. The core research 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 being developed. One specific aspect of management systems is their ability to handle erroneous software development 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.

3.3.6 Term Graph Rewriting

Term graph rewriting is a model for computing with graphs representing functional expressions. Graphs allow to share common subexpressions which improves the efficiency of conventional term rewriting in space and time. The article [Plu02] summarizes essentials of term graph rewriting concerning soundness, completeness, termination and confluence.

In [HP02c], the authors review the concept of term graph narrowing as an approach for solving equations by transformations on term graphs. Term graph narrowing combines term graph rewriting with first-order term unification. This mechanism is complete for all term rewriting systems over which term graph rewriting is normalizing and confluent. This includes, in particular, all convergent term rewriting systems. Completeness means that for every solution of a given equation, term graph narrowing can find an equivalent or more general solution. The general motivation for using term graphs instead of terms is to improve efficiency: sharing common subterms saves space and avoids the repetition of computations.

3.3.7 Application to Molecular Computing

Molecular Computing is an exciting, novel and fast growing research area rooted in computer science, but very interdisciplinary in nature (in particular, the interaction with molecular biology, biochemistry and nanotechnology is very vivid and successful). One of the interesting models of molecular computing are *Membrane Systems* which abstract from the rich membrane structure of living cells. Here, the membranes divide the cell space into separate regions. Each region has its own (computational) process going on, and the (process of the) regions can communicate through the membranes that separate/join them. The graph transformation methods turned out to be very suitable in the investigation of the computational nature of membrane systems. The results we have obtained are reported in [PR02, MPPR02].

References

[Bar02] R. Bardohl. A Visual Environment for Visual Languages. *Science of Computer Programming (SCP)*, 2002. To appear.

[BC02a] Michel Bauderon and Andrea Corradini, editors. *Proc. GETGRATS Closing Workshop*, volume 51 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, 2002.

[BC02b] P. Bottoni and G. Costagliola. On the definition of visual languages and their editors. In M. Hegarty, B. Meyer, and N.H. Narayanan, editors, *Proc. Diagrammatic Representation and Inference (DIAGRAMS 2002)*, volume 2317 of *Lecture Notes in Artificial Intelligence*, pages 305–319. Springer, 2002.

[BCK01] P. Baldan, A. Corradini, and B. König. A static analysis technique for graph transformation systems. In K.G. Larsen and M. Nielsen, editors, *Proc. CONCUR 2001*, volume 2154 of *Lecture Notes in Computer Science*, pages 381–395. Springer, 2001.

[BCK02] P. Baldan, A. Corradini, and B. König. Static analysis of distributed systems with mobility specified by graph grammars—a case study. In *Proc. IDPT*, 2002. To appear.

[BCM00] P. Baldan, A. Corradini, and U. Montanari. History preserving bisimulations for contextual nets. In D. Bert and C. Choppy, editors, *Proc. WADT'99*, volume 1827 of *Lecture Notes in Computer Science*, pages 291–310. Springer, 2000.

[BCM02] P. Baldan, A. Corradini, and U. Montanari. Bisimulation equivalences for graph grammars. In W. Brauer, H. Ehrig, J. Karhumäki, and A. Salomaa, editors, *Formal and Natural Computing - Essays Dedicated to Grzegorz Rozenberg [on occasion of his 60th birthday, March 14, 2002]*, volume 2300 of *Lecture Notes in Computer Science*, pages 158–190. Springer, 2002.

[BE01] R. Bardohl and C. Ermel. Visual Specification and Parsing of a Statechart Variant using GenGEd. In *Proc. Symposium on Visual Languages and Formal Methods (VLFM'01)*, Stresa, Italy, September 5–7 2001.

[BEE01] R. Bardohl, C. Ermel, and H. Ehrig. Generic Description of Syntax, Behavior and Animation of Visual Models. Technical Report 2001/19, Technische Universität Berlin, 2001. ISSN 1436-9915.

[BEE+02] R. Bardohl, K. Ehrig, C. Ermel, A. Qemali, and I. Weinhold. GenGEd – Specifying Visual Environments based on Visual Languages. In Kreowski and Knirsch [KK02], pages 71–82.

[BEH+02] B. Braatz, H. Ehrig, K. Hoffmann, J. Padberg, and M. Urbášek. Application of Graph Transformation Techniques to the Area of Petri Nets. In Kreowski and Knirsch [KK02], pages 35–43.

[BGRS01] R. Bardohl, M. Große-Rhode, and M. Simeoni. Interactive Rule-based Specification with an Application to Visual Language Definition. In *Proc. Int. Workshop on Algebraic Development Techniques (WADT'01)*, pages 3–4, April 2001.

[BH01] Giorgio Busatto and Berthold Hoffmann. Comparing notions of hierarchical graph transformation. In Baresi et al. [BPT01]. 8 pages.

[BHLM01] R. Bruni, F. Honsell, M. Lenisa, and M. Miculan. Comparing higher-order encodings in logical frameworks and tile logic. In U. Montanari, editor, *Proc. Final Workshop of the TOSCA Project (TOSCA 2001)*, volume 62 of *Electronic Notes in Theoretical Computer Science*, 2001.

[BJK02] Michel Bauderon, Hélène Jacquet, and Renate Klempien-Hinrichs. Pullback rewriting and applications. In Bauderon and Corradini [BC02a]. 10 pages.

[BKK01] Giorgio Busatto, Hans-Jörg Kreowski, and Sabine Kuske. Abstract hierarchical graph transformation. Technical Report 1/01, Universität Bremen, 2001.

[BKPT01] P. Bottoni, M. Koch, F. Parisi-Presicce, and G. Taentzer. A Visualization of OCL using Collaborations. In M. Gogolla and C. Kobryn, editors, *UML 2001 – The Unified Modeling Language*, volume 2185 of *Lecture Notes in Computer Science*, pages 257–271. Springer, 2001.

[BKPT02] P. Bottoni, M. Koch, F. Parisi-Presicce, and G. Taentzer. Working on OCL with Graph Transformation. In Kreowski and Knirsch [KK02], pages 1–10.

[BM01a] R. Bruni and U. Montanari. Transactions and zero-safe nets. In H. Ehrig, G. Juhás, J. Padberg, and G. Rozenberg, editors, *Advances in Petri Nets: Unifying Petri Nets*, volume 2128 of

Lecture Notes in Computer Science, pages 380–426. Springer, 2001.

[BM01b] R. Bruni and U. Montanari. Zero-safe net models for transactions in Linda. In U. Montanari and V. Sassone, editors, *Proc. Int. Workshop on Concurrency and Coordination (ConCoord 2001)*, volume 54 of *Electronic Notes in Theoretical Computer Science*, 2001.

[BM02] R. Bruni and U. Montanari. Dynamic connectors for concurrency. *Theoretical Computer Science*, 281(1-2):131–176, 2002.

[BMM02] R. Bruni, J. Meseguer, and U. Montanari. Symmetric monoidal and cartesian double categories as a semantic framework for tile logic. *Mathematical Structures in Computer Science*, 12(1):53–90, 2002.

[BMR01] R. Bruni, U. Montanari, and F. Rossi. An interactive semantics of logic programming. *Theory and Practice of Logic Programming*, 1(6):647–690, 2001.

[BPT01] Luciano Baresi, Mauro Pezzè, and Gabriele Taentzer, editors. *Proc. ICALP 2001 Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2001)*, volume 50.3 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, July 2001.

[BR01] R. Bardohl and L. Ribeiro. Towards a Compositional Approach to Define Graphical Animation of Software Applications. In *Proc. Symposium on Visual Languages and Formal Methods (VLFM'01)*, Stresa, Italy, September 5–7 2001.

[BST01] R. Bardohl, T. Schultzke, and G. Taentzer. Visual Language Parsing in GenGEd. In Baresi et al. [BPT01].

[BT01] P. Bottoni and G. Taentzer. Parsing and Semantics of a Statechart Variant by Contextual Layered Graph Transformation. In *Proc. Symposium on Visual Languages and Formal Methods (Statechart Modeling Contest), IEEE Symposia on Human-Centric Computing Languages and Environments (HCC'01)*, Stresa, Italy, 2001.

[DHK01a] R. Depke, R. Heckel, and J. M. Küster. Improving the agent-oriented modeling process with roles. In *Proc. 5th Int. Conference on Autonomous Agents (AGENTS-2001)*, Proc. of the ACM, pages 640–647, Montreal, Canada, June 2001. ACM.

[DHK01b] R. Depke, R. Heckel, and J. M. Küster. Roles in agent-oriented modeling. *Int. Journal of Software Engineering and Knowledge Engineering*, 11(3):281–302, 2001.

[DHM02] Frank Drewes, Berthold Hoffmann, and Mark Minas. Constructing shapely nested graph transformations. In Kreowski and Knirsch [KK02], pages 107–118.

[DHP02] Frank Drewes, Berthold Hoffmann, and Detlef Plump. Hierarchical graph transformation. *Journal of Computer and System Sciences*, 64(2):249–283, 2002.

[DJM02] S. Demeyer, D. Janssens, and T. Mens. Formalizing behaviour preserving program transformations. Technical Report 02–03, University of Antwerp - UIA, dept. of Math. and Comp. Sci., 2002.

[DK01a] Frank Drewes and Renate Klempien-Hinrichs. TREEBAG. In S. Yu and A. Păun, editors, *Proc. 5th Int. Conference on Implementation and Application of Automata (CIAA 2000)*, volume 2088 of *Lecture Notes in Computer Science*, pages 329–330. Springer, 2001.

[DK01b] Frank Drewes and Hans-Jörg Kreowski. Reading words in graphs generated by hyperedge replacement. In C. Martin-Vide and V. Mitrana, editors, *Where Mathematics, Computer Science, Linguistics and Biology Meet*, chapter 22, pages 243–252. Kluwer Academic Publishers, 2001.

[EBE01] C. Ermel, R. Bardohl, and H. Ehrig. Specification and Implementation of Animation Views for Petri Nets. In H. Weber, H. Ehrig, and W. Reisig, editors, *2nd Int. Colloquium on Petri Net Technologies for Modelling Communication Based Systems*, pages 75–92, Berlin, Germany, Sept. 2001. Researcher Group Petri Net Technology, Fraunhofer Gesellschaft ISST.

[EEP01] Hartmut Ehrig, Claudia Ermel, and Julia Padberg, editors. *Proc. Uniform Approaches to Graphical Process Specification Techniques (UNIGRA 2001), Satellite Workshop to ETAPS 2001*,

volume 44.4 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, 2001.

[EHK01] G. Engels, R. Heckel, and J.M. Küster. Rule-based specification of behavioral consistency based on the UML meta model. In M. Gogolla and C. Kobryn, editors, *Proc. UML 2001*, volume 2185 of *Lecture Notes in Computer Science*. Springer, 2001.

[EKGH01] G. Engels, J.M. Küster, L. Groenewegen, and R. Heckel. A methodology for specifying and analyzing consistency of object-oriented behavioral models. In V. Gruhn, editor, *Proc. European Software Engineering Conference (ESEC/FSE 01), Vienna, Austria*, volume 1301 of *Lecture Notes in Computer Science*, pages 327–343. Springer, 2001.

[EO01] H. Ehrig and F. Orejas. A generic component concept for integrated data type and process specification techniques. Technical Report 2001/12, Technische Universität Berlin, FB Informatik, 2001.

[EOB+02a] H. Ehrig, F. Orejas, B. Braatz, M. Klein, and M. Piirainen. A generic component concept for system modelling. In R. Kutsche and H. Weber, editors, *Proc. Formal Aspects of Software Engineering (FASE 2002)*, 2002.

[EOB+02b] H. Ehrig, F. Orejas, B. Braatz, M. Klein, and M. Piirainen. A transformation-based component framework for a generic integrated modeling technique. Technical Report 2002/2, ISSN 1436-9915, Technische Universität Berlin, FB Informatik, 2002.

[FMT01] G. Ferrari, U. Montanari, and E. Tuosto. A LTS Semantics of Ambients via Graph Synchronization with Mobility. In *Proc. ICTCS 2001*, volume 2202 of *Lecture Notes in Computer Science*. Springer, 2001.

[Gat01] Felix H. Gatzemeier. Authoring operations based on weighted schemata. In Guy W. Mineau, editor, *Conceptual Structures: Extracting and Representing Semantics — Contributions to ICCS 2001*, pages 61–74, August 2001.

[GH02] S. Gyapay and R. Heckel. Towards graph transformation with time. In Kreowski and Knirsch [KK02], pages 131–140.

[GJS02] M. Große-Rhode, S. John, and G. Schröter. Transformation Systems for the Integration of Software Specification. In Kreowski and Knirsch [KK02], pages 151–160.

[GM00] F. Gadducci and U. Montanari. The tile model. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, 2000. Also Technical Report TR-27/96, Dipartimento di Informatica, Università di Pisa, 1996.

[HH01] J.H. Hausmann and R. Heckel. Use cases as views: A formal approach to requirements engineering in the united process. In *Proc. Integrating Diagrammatic and Formal Specification Techniques, Tagungsband der GI/OCG-Jahrestagung*, pages 595–599, Wien, 2001. ISBN 3-85403-157-2.

[HHS01] J.H. Hausmann, R. Heckel, and S. Sauer. Towards dynamic meta modeling of UML extensions: An extensible semantics for UML sequence diagrams. In M. Minas and A. Schürr, editors, *Symposium on Visual Languages and Formal Methods, IEEE Symposia on Human Computer Interaction (HCI 2001), Stresa, Italy*, Los Alamitos, CA, September 2001. IEEE Computer Society Press.

[HHT02] J.H. Hausmann, R. Heckel, and G. Taentzer. Detection of conflicting functional requirements in a use case-driven approach. In *Proc. Int. Conference on Software Engineering (ICSE 2002)*, Orlando, FL, USA, 2002. To appear.

[HK01] R. Heckel and J.M. Küster. Behavioral constraints for visual models. In *Proc. ICALP2001 Workshop on Graph Transformation and Visual Modelling Techniques, Heraklion, Greece*, volume 50.3 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, 2001.

[HKT02] R. Heckel, J. Küster, and G. Taentzer. Towards automatic translation of UML models into semantic domains. In Kreowski and Knirsch [KK02], pages 11–22.

[HM01] Berthold Hoffmann and Mark Minas. Transformation of shaped nested graphs and diagrams. In M. van den Brand and R. Verma, editors, *Proc. 2nd Int. Workshop on Rule-Based Programming (RULE 2001)*, volume 59(4) of *Electronic Notes in Theoretical Computer Science*, 2001. 15 pages.

[HMP01] Annegret Habel, Jürgen Müller, and Detlef Plump. Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science*, 11(5):637–688, 2001.

[Hof01] Berthold Hoffmann. Shapely hierarchical graph transformation. In *Proc. IEEE Symposia on Human-Centric Computing Languages and Environments*, pages 30–37. IEEE Computer Press, 2001.

[HP02a] Annegret Habel and Detlef Plump. A core language for graph transformation. In Kreowski and Knirsch [KK02], pages 187–199.

[HP02b] Annegret Habel and Detlef Plump. Relabelling in graph transformation. In *Proc. Int. Conference on Graph Transformation (ICGT 2002)*, Lecture Notes in Computer Science. Springer, 2002. To appear.

[HP02c] Annegret Habel and Detlef Plump. Solving equations by graph transformation. In Bauderon and Corradini [BC02a].

[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*, volume 2185 of *Lecture Notes in Computer Science*. Springer, april 2001.

[HZ01] R. Heckel and A. Zündorf. How to specify a graph transformation approach: A meta model for FUJABA. In Ehrig et al. [EEP01].

[JV01] D. Janssens and N. Verlinden. Modeling Petri nets by local action systems. In Baresi et al. [BPT01].

[KBK01] Hans-Jörg Kreowski, Giorgio Busatto, and Sabine Kuske. GRACE as a unifying approach to graph-transformation-based specification. In Ehrig et al. [EEP01]. 15 pages.

[KBK+02] Hans-Jörg Kreowski, Giorgio Busatto, Renate Klempien-Hinrichs, Peter Knirsch, and Sabine Kuske. Structured modeling with GRACE. In Bauderon and Corradini [BC02a]. 13 pages.

[KK02] Hans-Jörg Kreowski and Peter Knirsch, editors. *Proc. APPLIGRAPH Workshop on Applied Graph Transformation (AGT 2002)*, ETAPS 2002 Satellite Events. University of Grenoble, 2002.

[KKK02] Renate Klempien-Hinrichs, Peter Knirsch, and Sabine Kuske. Modeling the pickup-and-delivery problem with structured graph transformation. In Kreowski and Knirsch [KK02], pages 119–130.

[Kle01] Renate Klempien-Hinrichs. *Context-free Hypergraph Grammars. Node and Hyperedge Rewriting with an Application to Petri Nets.* Books on Demand, Norderstedt, 2001. Doctoral dissertation, Universität Bremen, 2000.

[Kle02a] Renate Klempien-Hinrichs. Context-free hypergraph grammars with node rewriting. In Bauderon and Corradini [BC02a]. 10 pages.

[Kle02b] Renate Klempien-Hinrichs. Normal forms for context-free node-rewriting hypergraph grammars. *Mathematical Structures in Computer Science*, 12:135–148, 2002.

[KM01] B. König and U. Montanari. Observational equivalence for synchronized graph rewriting. In N. Kobayashi and B. Pierce, editors, *Proc. TACS 2001*, volume 2215 of *Lecture Notes in Computer Science*, pages 145–164. Springer, 2001.

[KMP00] 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. 6th European Symposium on Research in Computer Security (ESORICS 2000)*, volume 1895 of *Lecture Notes in Computer Science*, pages 122–139. Springer, 2000.

[KMP01a] 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 *Lecture Notes in Computer Science*, pages 287–302. Springer, March 2001.

[KMP01b] M. Koch, L.V. Mancini, and F. Parisi-Presicce. On the specification and evolution of access control policies. In S. Osborn, editor, *Proc. 6th ACM Symp. on Access Control Models and Technologies*, pages 121–130. ACM, May 2001.

[KMP02] M. Koch, L. V. Mancini, and F. Parisi-Presicce. A Framework for Role-Based Access Control. *ACM Transactions on Information and System Security (TISSEC)*, August 2002. To appear.

[Kre02] Hans-Jörg Kreowski. A sight-seeing tour of the computational landscape of graph transformation. In W. Brauer, H. Ehrig, J. Karhumäki, and A. Salomaa, editors, *Formal and Natural Computing. Essays Dedicated to Grzegorz Rozenberg*, volume 2300 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2002.

[Kus01] Sabine Kuske. A formal semantics of UML state machines based on structured graph transformation. In Martin Gogolla and Cris Kobryn, editors, *UML 2001 – The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, volume 2185 of *Lecture Notes in Computer Science*, pages 241–256, 2001.

[Kus02] Sabine Kuske. Parameterized transformation units. In Bauderon and Corradini [BC02a]. 12 pages.

[MH01a] André Marburger and Dominikus Herzberg. E-CARES research project: Understanding complex legacy telecommunication systems. In *Proc. 5th European Conference on Software Maintenance and Reengineering*, pages 139–147, Lisbon, Portugal, 2001. IEEE Computer Society Press.

[MH01b] Mark Minas and Berthold Hoffmann. Specifying and implementing visual process modeling languages with DiaGen. In Ehrig et al. [EEP01]. 13 pages.

[MH02] André Marburger and Dominikus Herzberg. E-CARES Research Project: Utilizing Dynamic Information. In *Proc. 3rd Workshop Software Reengineering (WSR 2001)*, pages 47–49. Fachberichte Informatik Universität Koblenz, 2002.

[MPPR02] C. Martin-Vide, A. Păun, G. Păun, and G. Rozenberg. Membrane systems with coupled transport: Universality and normal forms. *Fundamenta Informaticae*, 49:1–15, 2002.

[Plu02] Detlef Plump. Essentials of term graph rewriting. In Bauderon and Corradini [BC02a].

[PR02] G. Păun and G. Rozenberg. A guide to membrane computing. *Theoretical Computer Science*, 2002. To appear.

[Tae01] G. Taentzer. Towards common exchange formats for graphs and graph transformation systems. In Ehrig et al. [EEP01].

[Tae02a] G. Taentzer. A Visual Modeling Framework for Distributed Object Computing. In B. Jacobs and A. Rensink, editors, *Proc. Formal methods for open object-based distributed systems V (FMODS'02)*. Kluwer Academic Publishers, 2002.

[Tae02b] G. Taentzer. Visual modeling of distributed object systems by graph transformation. In Bauderon and Corradini [BC02a].

[TKH+01] I.J. Timm, P. Knirsch, O. Herzog, H.K. Tönshoff, and P.-O. Woelk. Mass Customization als Chance für KMU – Kooperative Agenten für die Informationslogistik. In H.-J. Sebastian and T. Grünert, editors, *Logistik Management – Supply Chain Management und e-Business, Aachen*, pages 401–409. B.G. Teubner Verlag, 2001. In German.

[TWK+01] I.J. Timm, P.-O. Woelk, P. Knirsch, H.K. Tönshoff, and O. Herzog. Flexible mass customisation: Managing its information logistics using adaptive co-operative multiagent systems. In K.S. Pawar and M. Muffatto, editors, *Logistics and the Digital Economy. Proc. 6th Int. Symposium on Logistics, Salzburg, Austria*, pages 227–232, 2001.

[Ver02] N. Verlinden. *Process Semantics for Embedding-Based Graph Transformation Systems.* PhD thesis, University of Antwerp - UIA, dept. of Math. and Comp. Sci., 2002.