Application of Graph Transformation Techniques to the Area of Petri Nets: An Overview *

Extended Abstract

B. Braatz, H. Ehrig, K. Hoffmann, J. Padberg, M. Urbášek

Technical University Berlin, Germany
Institute for Software Technology and Theoretical Computer Science
{bbraatz, ehrig, hoffmann, padberg, urbasek}@cs.tu-berlin.de

1 Aims and Introduction

The main aim of this contribution 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.

Since about 10 years the strong relationships between the areas of Petri nets and graph transformation systems have been studied especially as part of the cooperation of the groups in Pisa and Berlin, while it was observed by Kreowski already in the early 80s how Petri nets can be considered as a special case of graph transformation systems. The main aim of the Pisa-Berlin cooperation was to transfer the well-known constructions leading to a truly concurrent event structure semantics from Petri nets to graph transformation systems. Vice versa the concept of high-level replacement systems, short HLR-systems, by Ehrig, Habel, Kreowski and Parisi-Presicce in [EHKP91] was the starting point to obtain new concepts and results for the area of Petri nets by application of graph transformation techniques. In fact the concept of HLR-systems in [EHKP91] is a generalization of the double pushout approach from graph transformation to HLR-systems in a categorical framework.

The instantiation of HLR-systems to Petri nets leads to the concept of net transformation systems [PER95], which will be discussed in Section 2. In order to study property preserving transformations, the concept of HLR-transformations was extended by Padberg in [Pad96] to $\mathcal{Q}$-transformations leading to safety and liveness preserving transformations in [PGE98,GHP99] and [GPU01] respectively reviewed in Section 3. In Section 4 we discuss in which way the module concept for graph transformation systems developed by Simeoni [Sim00] has been transferred to Petri nets [PHBS02] and to a generic component concept in [EOBKP02]. In Section 5 we discuss the modeling of open systems, where the

* This work is partially supported by the project APPLIGRAPH (ESPRIT Basic Research WG), GRAPHIT (CNPq and DLR) and by the joint research project "DFG-Forschergruppe Petrinetz-Technologie" between H. Weber (Coordinator), H. Ehrig (both from the Technical University Berlin) and W. Reisig (Humboldt University Berlin), supported by the German Research Council (DFG).

concept of open graph transformation systems developed by Heckel [Hec98] has influenced the development of open nets in [BCEH01]. Finally in Section 6 we briefly mention other topics, where the areas of graph transformation techniques and Petri nets have influenced each other.

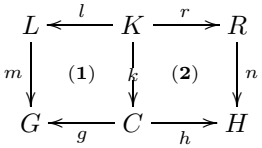
2 Net Transformation Systems as Instantiation of High-Level Replacement Systems

The general idea of high-level replacement (HLR) systems is to generalize the concept of graph transformation systems and graph grammars from graphs to all kinds of structures which are of interest [EHKP91]. This generalization has been done categorically and can be applied to all kinds of high-level structures, especially also different kinds of Petri nets. Several results from graph grammars have been reformulated in the framework of high-level replacement systems and can be applied to other high-level structures without the necessity to be proven again. The theory of HLR systems is based on the double pushout approach, which has been widely investigated in the area of graph grammars (see [Ehr79]). The HLR framework is suitable for many high-level structures. The concept of transformations has been applied to several classes of Petri nets (P/T Petri nets, colored Petri nets, AHL nets), yielding the idea of net transformation systems first introduced in [PER95].

The next definition introduces rules, transformations and net transformation systems formally for a given category **NET** of low or high level nets. More about the underlying theory can be found in [PER95] and in [Pad99].

Definition (Rules, Transformations and Transformation Systems).

1. *A* rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ *in a category* **NET** *consists of the objects* L, K *and* R, *called left-hand side, interface (or gluing object), and right-hand side, respectively, and two morphisms* $K \xrightarrow{l} L$ *and* $K \xrightarrow{r} R$ *with both morphisms* $l, r \in \mathcal{M}$, *a suitable class of injective morphisms in* **NET**.
2. *Given a rule* $p = (L \xleftarrow{l} K \xrightarrow{r} R)$, *a* direct transformation $G \overset{p}{\Longrightarrow} H$ *from a net* G *to a net* H *is given by two pushout diagrams* **(1)** *and* **(2)** *in the category* **NET** *as shown below.*



The morphisms $L \xrightarrow{m} G$ *and* $R \xrightarrow{n} H$ *are called occurrences of* L *in* G *and* R *in* H, *respectively. By an occurrence of rule* $r = (L \xleftarrow{l} K \xrightarrow{r} R)$ *in a net* G *we mean an occurrence of the left-hand side* L *in* G.
In fact, the occurrence morphism m *has to satisfy a specific condition, called gluing condition, in order to apply the rule* p *to the net* G.

3. *Given a category of nets* **NET** *together with a suitable class of injective morphisms* $\mathcal{M}$*, a* net transformation system $H = (\mathcal{S}, \mathcal{P})$ *in* $(\mathbf{NET}, \mathcal{M})$ *is given by a start net* $\mathcal{S} \in |\mathbf{NET}|$*, and a set of rules* $\mathcal{P}$*.*

The idea of net transformation systems is the basic idea behind the stepwise development of communication based systems in the framework of Petri nets. Each transformation step can be formally depicted as a rule-based transformation according to an appropriate rule in a specific net transformation system. The transformation sequence then provides a transformation from the initial net G to the final net H as shown below.

$$G = G_0 \stackrel{p_1}{\Longrightarrow} G_1 \stackrel{p_2}{\Longrightarrow} \ldots \stackrel{p_n}{\Longrightarrow} G_n = H.$$

Several results concerning horizontal structuring for net transformation systems have been adopted from HLR systems. The two basic structuring constructions for nets are union and fusion. Union allows a construction of larger nets from smaller ones with shared subpart, while fusion is a construction which allows to identify distinguished subnets.

Especially the concept of fusion is more general then the concept of fusion of places often cited in the Petri net literature. The fusion introduced in net transformation systems is not restricted to the fusion of places, but covers also fusion of subnets.

In a view of software development methodology it is important that horizontal structuring based on fusion and union is compatible with transformations. This means that – under certain compatibility conditions – the result is the same whether we apply first union/fusion and then transformation or vice versa. For more details about horizontal structuring and transformations see e.g. [Pad96,Pad99].

This net transformation technique including horizontal structuring has been successfully applied in the case study of a larger medical information system summarized in [EPE96].

3 Property Preserving Net Transformations

Although the net transformation framework is a suitable concept for stepwise development of systems, very often there is a need to consider in addition more general morphisms for refinement or abstraction. The main idea is to enlarge the category of nets by $\mathcal{Q}$-morphisms in the sense of [Pad96] in order to formulate refinement/abstraction morphisms.

More precisely, another category of nets **QNET** with a distinguished class of morphisms $\mathcal{Q}$, called $\mathcal{Q}$-morphisms, is employed. The category **NET** of nets from the previous definition in Section 2 has to be a subcategory of **QNET**. The class of $\mathcal{Q}$-morphisms is the class of refinement/abstraction morphisms. This class of morphisms has to satisfy additional requirements called $\mathcal{Q}$-conditions (see [Pad96]) to be adequate for refinement or abstraction.

Then, a single transformation step is formally given by Fig. 1. The squares **(1)** and **(2)** represent a transformation in the category **NET** (as in definition in Section 2). A morphism q is the refinement/abstraction morphism in **QNET**, such that $q \in \mathcal{Q}$. When the $\mathcal{Q}$-conditions are satisfied then there exists an induced $\mathcal{Q}$-morphism $q' \in \mathcal{Q}$ in **QNET**, which is a morphism between the original and the transformed net.

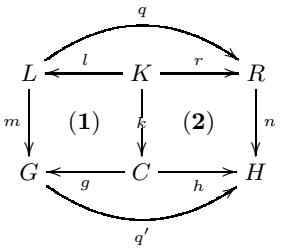


Fig. 1. Transformation step

The idea of $\mathcal{Q}$-morphisms for net transformation systems has also been introduced for HLR systems in general. It is a powerful formal technique for design of complex systems. However, it has been applied up to now only in the domain of Petri nets. The application to graph transformation systems is certainly also of interest and a point of future investigation.

The concept of $\mathcal{Q}$-morphisms is important in order to study whether the transformation of nets is property preserving. During the transformation process, a net may become too large to check some properties of this net efficiently. If this property could be checked or stated for an initial net before the transformation starts and then preserved during the transformation process, a tedious investigation of properties for the final net can be omitted.

The idea of property preserving transformations has been investigated in [GHP99,PG00,PGE98] for safety properties and in [GPU01] for liveness. Safety-properties for Petri nets are stated as propositional logic formulas upon the actual marking of Petri nets. Morphisms preserving safety properties have been investigated for several types of Petri nets, see [GHP99] for P/T Petri nets, [PG00] for colored Petri nets and [PGE98] for a class of algebraic-high level nets. This class of morphisms has been applied also in a case study of a medical information system in [Pad99] in order to prove relevant properties of the information system.

Liveness preserving refinement is based on the standard notion of liveness as used in Petri net theory. Liveness means that no deadlock or even livelock can occure. In [GPU01] it is shown that a special type of transition refinement preserves liveness in Petri nets. The idea is based on abstracting morphisms, which are related to vicinity respecting morphisms (introduced in [DM90]). A certain subclass of abstracting morphisms, called a class of collapsing morphisms,

allows a description of a transition refinement as collapsing of a subnet to one transition. The preservation of liveness has been proven for collapsing morphisms and demonstrated on an example in [GPU01].

Both types of property preserving transformations give an oportunity to cut the cost of verification of system properties for large systems.

4 Module and Component Concepts

A variety of modularity concepts has been introduced for graph transformation systems in the literature. An overview and classification is given in [HEET99]. In the following we consider the concept of M. Simeoni introduced in [Sim00] which has been adapted also to Petri nets.

Modules of graph transformation systems in the sense of [Sim00] are based on refinement morphisms exp between export EXP and body BOD of a module and injections imp between the import IMP and the body (cf. Fig. 2). It is important to notice that IMP, EXP and BOD are graph transformation systems rather than single graphs. The refinement morphism represents the elaboration of certain aspects of the graph transformation system in the export interface to greater detail in the body. The import interface contains the parts of the body to be further refined.

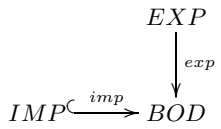


Fig. 2. Module with exp-refinement and imp-inclusion

It is shown in [Sim00] that the category consisting of typed algebraic graph transformation systems as objects and refinements as morphisms has pushouts if at least one of the morphisms is an inclusion. This means that refinements of the import of a module yield corresponding refinements of the body.

In [PHBS02] this approach is applied to Petri nets leading to Petri net modules. In this case IMP, EXP and BOD are Petri nets. Refinement morphisms are defined as a more general kind of Petri net morphisms allowing transitions not only to be mapped to transitions, but also to whole subnets of the target net. As in the case of graph transformation systems composition of modules can be defined by lifting of a refinement from the import to the body.

Abstracting the ideas in [Sim00] and [PHBS02] a concept of components for generic modeling techniques is introduced in [EOBKP02], which allows arbitrary kinds of transformations between the export and the body of a component. The only property that must be satisfied by the chosen transformation concept is the extension property, which requires that transformations can be lifted along an inclusion.

Moreover, a transformation semantics is proposed which is suitable for semi-formal modeling techniques like the UML, as well as for formal modeling techniques with tight semantics like Petri nets. This semantics is a function, yielding for every transformation of the import of a component a corresponding transformation of the export. The transformation semantics is shown to be compositional, i. e. the semantics of a composed component can be obtained from the semantics of its parts. These results of [EOBKP02] can be transferred back to Petri nets as well as graph transformation systems.

5 Modeling of Open Systems

In the double-pushout (DPO) approach, a graph derivation $G \overset{p,m}{\Longrightarrow} H$ is uniquely determined up to isomorphism for a rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ by an occurence morphism $m\colon L \rightarrow G$ and the requirement that **(1)** and **(2)** in Fig. 3 are pushout diagrams. Similar to other graph transformation systems this can be considered as a tight semantics assigned to rules.

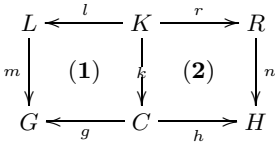


Fig. 3. Double pushout resp. double pullback diagram

The modeling of open systems, however, requires a kind of loose semantics, because in addition to the changes specified by the rules, unspecified changes can occur due to interaction of the system with the environment or a user. In [Hec98] this loose semantics is achieved by a double-pullback (DPB) approach leading to graph transitions $G \overset{p/d}{\longrightarrow} H$. In contrast to the DPO approach, the requirement that **(1)** and **(2)** in Fig. 3 are pullback diagrams leaves changes to the environment of the occurence unspecified. Therefore a graph transition is not uniquely defined by a morphism $m\colon L \rightarrow G$ but only by a triple of morphisms $d = \langle d_L, d_K, d_R \rangle$, determining the implicit changes to the environment.

A second important concept of [Hec98] is the explicit frame condition, restricting the parts of the graph on which interaction with the environment can take place to output and input sorts O^-, O^+ and requiring transitions to be direct derivations for the remaining parts. This approach is applied to Petri nets by the notion of open Petri nets [PJHE98]. An open net consists of an ordinary Petri net N and sets of output and input places O^-, O^+, on which tokens can be exchanged with the environment. This concept has been formally introduced in [BCEH01] where open nets are used to model two workflows interacting with each other by a common interface. The open places in either of the nets for the

two workflows represent the environment through which communication with the other one is possible.

As a main result in [BCEH01] a Goltz-Reisig process semantics is introduced for open nets. For these processes an amalgamation much like the amalgamation of algebras (see e. g. [EM85]) is developed, which can be used to achieve compositionality of the semantics of open nets. It seems promising to transfer these results back to graph transformation systems. This, however, is future work.

The concept of inclompete information in open systems leads to the framework presented in [EHLOPR00] for generic rule-based modeling techniques, which is instantiated on one hand by the graph transitions mentioned above, on the other hand by open Petri nets, where rules correspond to net transitions $t{:}\, pre(t) \Rightarrow post(t)$, transformations to the firing of a net transition and transitions to the firing of a net transition with unspecified output and input on the open places.

6 Further Recent Developments

In [Hof00,Hof01] we have introduced the concept and formal definition of Algebraic Higher Order Nets where the data type part is extended by higher order types, sorts and functions. This allows to have functions as data items on places, such that different functions may be activated and applied during run time. As far as we can see it is possible to extend analogously the data type part of attributed graph transformation leading to higher order attributed graph transformation. As part of ongoing work we will describe on one hand the theory of higher order attributed graph transformation and on the other hand the application domain where this feature of flexible modeling is especially useful.

A further recent development based on transformations for arbitrary specification techniques in the sense of high-level replacement systems is the description of architecture evolution. The architecture of specifications is represented by a graph that is used as a diagram over the specification. These diagram functors can then be transformed in the usual double-pushout approach. The formal foundation is given in [Pad01]. An informal version where an extensive example and its implementation in GenGEd is discussed can be found in [BEQ02].

7 Conclusion

In this paper we have presented four main topics where techniques from the area of graph transformation systems have influenced new developments in the area of Petri nets. The most promising ones for further development are the property preserving transformations in Section 3 and the module and component concepts in Section 4, because they support stepwise construction and verification of large systems from small components in the framework of Petri nets. An additional topic discussed in another overview paper [BEEQW02] are visual modeling techniques based on attributed graph transformation systems [Bar99]

which have been applied to Petri nets leading to an interesting new concept of animation of Petri nets [EBE01,BEEQW02].

References

[Bar99] R. Bardohl. *Visual Definition of Visual Languages based on Algebraic Graph Transformation*. PhD thesis, Technische Universität Berlin, 1999. Published by Verlag Dr. Kovac, 2000.

[BCEH01] P. Baldan, A. Corradini, H. Ehrig, and R. Heckel. Compositional Modeling of Reactive Systems Using Open Nets. In *Proc. CONCUR 2001*, Springer LNCS 2154, pages 502–518. Springer Verlag, 2001.

[BEEQW02] R. Bardohl, K. Ehrig, C. Ermel, A. Qemali, and I. Weinhold. Specifying Visual Languages with GenGEd. In *Proc. AGT 2002: APPLIGRAPH Workshop on Applied Graph Transformation*. 2002.

[BEQ02] R. Bardohl, C. Ermel, and A. Qemali. Transforming Specification Architectures with GenGEd. Submitted, 2002.

[DM90] J. Desel and A. Merceron. Vicinity Respecting Net Morphisms. In *Advances in Petri Nets*, Springer LNCS 483, pages 165–185. Springer Verlag, 1990.

[EBE01] C. Ermel, R. Bardohl, and H. Ehrig. Specification and Implementation of Animation Views for Petri Nets. In *Proc. 2nd Int. Colloquium on Petri Net Technologies for Modelling Communication Based Systems*, pages 75–92. 2001.

[EHKP91] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. From Graph Grammars to high level replacement systems. In *Proc. 4th Int. Workshop on Graph Grammars and their Application to Computer Science*, Springer LNCS 532, pages 269–291. Springer Verlag, 1991.

[EHLOPR00] H. Ehrig, R. Heckel, M. Llabrés, F. Orejas, J. Padberg, and G. Rozenberg. Double-Pullback Graph Transitions: A Rule-Based Framework with Incomplete Information. In *Proc. 6th Int. Workshop on Theory and Application of Graph Transformation*, Springer LNCS 1764, pages 85–102. Springer Verlag, 2000.

[Ehr79] H. Ehrig. Introduction to the Algebraic Theory of Graph Grammars. In *Proc. 1st Graph Grammar Workshop*, Springer LNCS 73, pages 1–69. Springer Verlag, 1979.

[EM85] H. Ehrig, B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, Vol. 6 of *EATCS Monographs on Theor. Comp. Science*. Springer Verlag, 1985.

[EOBKP02] H. Ehrig, F. Orejas, B. Braatz, M. Klein, and M. Piirainen. A Generic Component Concept for System Modeling. In *Proc. FASE '02*. 2002.

[EPE96] C. Ermel, J. Padberg, and H. Ehrig. Requirements Engineering of a Medical Information System Using Rule-Based Refinement of Petri Nets. In *Proc. IDPT '96*, pages 186–193. 1996.

[GHP99] M. Gajewsky, K. Hoffmann, and J. Padberg. Place Preserving and Transition Gluing Morphisms in Rule-Based Refinement of Place/Transition Systems. Technical Report 1999-14. Technical University Berlin, 1999.

[GPU01] M. Gajewsky, J. Padberg, and M. Urbášek. Rule-Based Refinement for Place/Transition Systems: Preserving Liveness-Properties. Technical Report 2001-8. Technical University Berlin, 2001.

[Hec98] R. Heckel. *Open Graph Transformation Systems: A New Approach to the Compositional Modelling of Concurrent and Reactive Systems.* PhD thesis, Technische Universität Berlin, 1998.

[HEET99] R. Heckel, G. Engels, H. Ehrig, and G. Taentzer. Classification and Comparison of Modularity Concepts for Graph Transformation Systems. In *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 2: Applications, Languages and Tools.* World Scientific, 1999.

[Hof00] K. Hoffmann. Run Time Modification of Algebraic High Level Nets and Algebraic Higher Order Nets using Folding and Unfolding Construction. In *Proc. 3rd Int. Workshop Communication Based Systems*, pages 55–72. 2000.

[Hof01] K. Hoffmann. Flexible Modellierung mit Algebraischen Higher Order Netzen. In *Proc. Workshop Modellierung*, pages 101–110. 2001.

[Pad96] J. Padberg. *Abstract Petri Nets: A Uniform Approach and Rule-Based Refinement.* PhD thesis, Technische Universität Berlin, 1996. Published by Shaker Verlag.

[Pad99] J. Padberg. Categorical Approach to Horizontal Structuring and Refinement of High-Level Replacement Systems. In *Applied Categorical Structures*, 7(4):371–403. 1999.

[Pad01] J. Padberg. *Formal Foundation for Transformations of Specification Architectures.* Technical Report. Technical University Berlin, 2001.

[PER95] J. Padberg, H. Ehrig, and L. Ribeiro. Algebraic High-Level Net Transformation Systems. In *Mathematical Structures in Computer Science*, 2:217–256. 1995.

[PG00] J. Padberg and M. Gajewsky. Safety Preserving Transformations of Coloured Petri Nets. Technical Report 2000-13. Technical University Berlin, 2000.

[PGE98] J. Padberg, M. Gajewsky, and C. Ermel. Rule-Based Refinement of High-Level Nets Preserving Safety Properties. In *Proc. FASE '98*, Springer LNCS 1382, pages 221–238. Springer Verlag, 1998.

[PHBS02] J. Padberg, K. Hoffmann, M. Buder, and A. Sünbül. Petri Net Modules for Component-Based Software Engineering. Technical Report. Technical University Berlin, 2002.

[PJHE98] J. Padberg, L. Jansen, R. Heckel, and H. Ehrig. Interoperability in Train Control Systems: Specification of Scenarios using Open Nets. In *Proc. IDPT '98*, pages 17–28. Society for Design and Process Science, 1998.

[Sim00] M. Simeoni. *A Categorical Approach to Modularization of Graph Transformation Systems using Refinements.* PhD thesis, Università di Roma La Sapienza, 2000.