Journal of Computer and Communications, 2015, 3, 20-28
Published Online February 2015 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2015.32002

Scientific Research Publishing

A Survey of Control Structures for Reconfigurable Petri Nets

Julia Padberg, Kathrin Hoffmann

Hamburg University of Applied Science, Hamburg, Germany
Email: julia.padberg@haw-hamburg.de

Received 2 February 2015; accepted 20 February 2015; published 27 February 2015

Abstract

Software systems are increasingly executed in dynamic infrastructures. These infrastructures are dynamic as they are themselves subject to change as they support various applications that may or may not share some of the resources. Dynamic software systems become more and more important, but are difficult to handle. Modeling and simulating dynamic systems requires the representation of their processes and the system changes within one model. To that effect, reconfigurable Petri nets consist of a Petri net and a set of rules that can modify the Petri net. Their main feature is the capability to model complex coordination behavior in dynamically adapting infrastructures. The interplay of both levels of dynamic behavior requires a very precise description, so the specification when and which rules are to be applied plays a crucial role for the convenient use of reconfigurable nets. We differentiate several types of reconfigurable Petri nets and present a survey of control structure for these types, reconfigurable Petri nets. These control structures either concern the infrastructure, *i.e.*, the rules and transformations or the system part, *i.e.*, the firing behavior, or both. They are introduced by a short characterization and illustrated by examples. We state the results for various Petri net types and the tools supporting the different control structures.

Keywords

Reconfigurable Petri Nets, Net Transformation, Control Structures, Model Transformation, Dynamic Software Systems

1. Introduction

In the field of systems engineering modeling plays a key role for understanding and controlling the behavior of the corresponding systems. Software systems are increasingly characterized by dynamic structures that require

How to cite this paper: Padberg, J. and Hoffmann, K. (2015) A Survey of Control Structures for Reconfigurable Petri Nets. *Journal of Computer and Communications*, **3**, 20-28. http://dx.doi.org/10.4236/jcc.2015.32002

execution and reconfiguration at run-time to adjust the systems behavior to its changing environment. Their main feature results from their complex coordination behavior within dynamically adapting infrastructures. Such dynamic structures need a suitable formal description technique that allows the separation at different levels of dynamic behavior within one model.

Reconfigurable Petri nets provide dynamic changes at the process level, (as typical for Petri nets) and additionally at the structure level. They are based on the algebraic approach to Petri nets, with operations describing the pre- and post-domain of transitions and are equipped with rules for the transformation of the net. These rules allow the modification of the net's structure at run time. Reconfigurable Petri nets have been applied in various application areas where complex coordination and structural adaptation at run-time is required (e.g. mobile *ad-hoc* networks [1], communication spaces [2] [3], ubiquitous computing [4] [5], concurrent systems [6], workflows in a dynamic infrastructure [7]). The distinction between the net behavior and the dynamic change of its net structure is the characteristic feature that makes reconfigurable Petri nets so suitable for systems with dynamic structures.

In **Figure 1** a screenshot of the tool ReConNet (see [8]) is given, depicting a decorated PT net $N1$ that can be modified using the rule $r: L \to R$, replacing a left hand side L by a right hand side R. The net can initially fire only once. After applying the rule r, that reverses the arcs, the resulting net is then live. Subsequent transformation steps yield alternatingly a net that is live or one that has a deadlock.

Several case studies have shown the advantages of reconfigurable Petri nets. In [5] reconfigurable Petri nets based on decorated place/transition nets have been used to model and analyze scenarios in a smart home. The tenant's behavior follows specific procedures that the smart home has to support. These procedures depend on the situation as well as the available sensor data and the tenant's action. These procedures are captured informally as scenarios and have been modeled formally for the better understanding of the possible interaction of the smart home and its tenant. There the scenarios describe the tenant's procedures and the transformations describe the dynamic change of the infrastructure as the reactions to the tenant's actions that both can be adequately captured.

Algebraic High-Level (AHL) nets are Petri nets combined with algebraic specifications [9] leading to the concept of algebraic high-level nets with suitable composition results. A general modeling framework for communication platforms and scenarios has been presented in [3] [10] using reconfigurable AHL nets. This framework employs an integration of Petri nets, algebraic data types and net transformation techniques. It allows the analysis of the evolution of communication platforms, the analysis of scenario evolutions and the investigation of user interactions on communication platforms. Reconfigurable AHL nets have also been used in [2] [11] for a case study on modeling a concrete communication platform—namely Skype. The behavior of the Skype clients has been modeled in detail and the whole system specification has been demonstrated for concrete use case scenarios. For these scenarios model properties have been formulated and validated. Ubiquitous computing systems

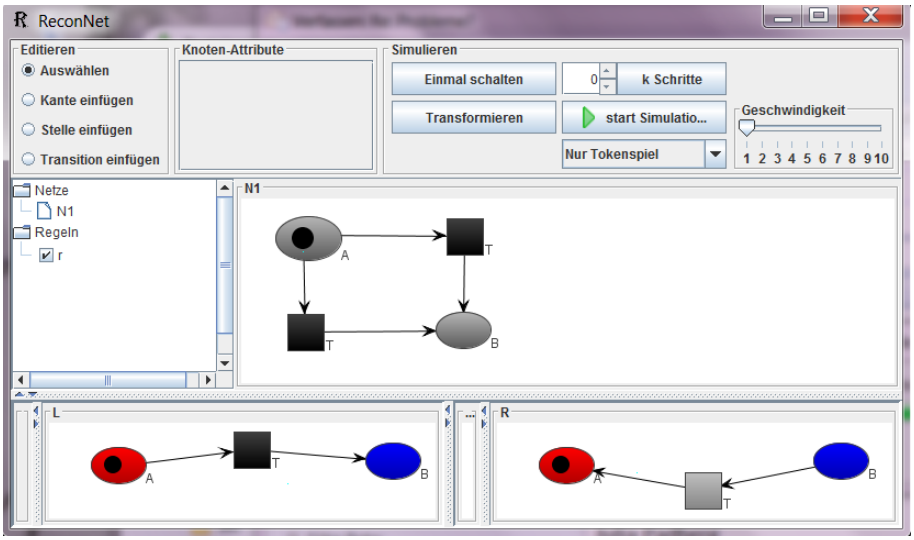

Figure 1. Reconfigurable Petri net with $(N, \{r\})$.

(UCSs) start to penetrate almost imperceptibly in everyday life. To ensure a solid operation, a UCS needs reliable and efficient communication between its distributed computing components. [4] presents a formal approach based on reconfigurable algebraic higher order nets with individual tokens (AHOI) nets [12]. This approach allows modeling the synchronous and asynchronous communication in UCSs. AHOI nets contain nets and transformation rules as individual tokens.

In this paper we give an overview of possible control structures for reconfigurable Petri nets. Moreover, we introduce two new control structures for decorated reconfigurable nets, namely net transformation units and rule priorities. Moreover, we discuss those control structures for reconfigurable nets that have been already introduced. In Section 2 we explain reconfigurable Petri nets and give an introductory example that we use in the subsequent section. Next we introduce two new control structures for reconfigurable Petri nets. In Section 3.1 we define net transformation units and state their main features. In Section 3.2 we introduce application conditions. First we examine negative application conditions and next we discuss nested application conditions. Further control structures are investigated subsequently. We summarize in Section 4 the investigated control structures and point out related work. The last section concerns future work and concluding remarks.

2. Basic Notions

In the algebraic approach to Petri nets a place/transition net is given by $N = (P, T, pre, post, m)$ with pre- and post-domain functions $pre, post : T \to P^{\oplus}$ and a marking $m \in P^{\oplus}$, where $P^{\oplus}$ is the multiset of places. A transition t is m-enabled for a marking if we have $pre(t) \le m$, and in this case the follower marking is given by $m' = (m - pre(t)) + post(t)$ and $m[t > m'$ is called a firing step. In [13] new features have been added to gain an adequate modeling technique. The extension to capacities and names is quite obvious. More interesting are the transition labels that may change, when the transition is fired. This allows a better coordination of transition firing and rule application, for example can be ensured that a transition has fired (repeatedly) before a transformation may take place. This last extension is conservative with respect to Petri nets as it does not change the net behavior, but it is crucial for the coordination of rule application and transition firing.

Reconfigurable Petri nets exhibit dynamic behavior by the token game of Petri nets and by net transformation by rule application. The transformation concept that is used for reconfigurable Petri nets is double-pushout approach on directed, labeled graphs. This approach has been lifted to a categorical framework using a morphism class $\mathcal{M}$, with various instantiations, called $\mathcal{M}$-adhesive high-level replacement (HLR) systems (see [14]). Place/transition nets (see [15]) as well as decorated place/transition nets (see [13]) and AHL nets (see [16]) have been shown to be $\mathcal{M}$-adhesive HLR categories. So they conform to a categorically defined replacement systems, that provides the main constructions as well as a huge amount of notions and results (see [15]). Other net types, e.g., elementary nets, colored nets, have been shown to have that properties as well, but have not been further investigated.

Net rules and transformations are based on net morphisms. These map places to places and transitions to transitions. They are given as a pair of mappings for the places and the transitions, so that the structure and the decoration are preserved and the marking may be mapped strictly. A rule is given by the left-hand side, interface and right-hand side net. In this contribution we omit the interface due to reasons of space and concentrate on the essence of net rules and transformations. An occurrence morphism o identifies the left-hand side in the given net N. Then the application of a rule r yields a transformation step $N \overset{r}{\Rightarrow} N'$ can be constructed in two steps, provided that specific gluing conditions hold. The characterization of specific points is a sufficient condition for the existence and uniqueness of the transformation.

The combination of one net together with a set of rules leads to a reconfigurable Petri nets $RN = (N, R)$, where N is a Petri net, more precisely a place/transition net, a decorated place/transition net or an AHL net and R is a set of net rules of the corresponding net type.

In **Figure 2** the transformation of the net N by applying the rule r is depicted, where the grey transition with label T is replaced by a transition with reversed arcs. Obviously, N' is now enabled and live.

3. Control Structures

Modeling with reconfigurable nets is obviously Turing complete, but to be practically relevant, control structures are required to specify the application of the rules more precisely. These control structures may belong to

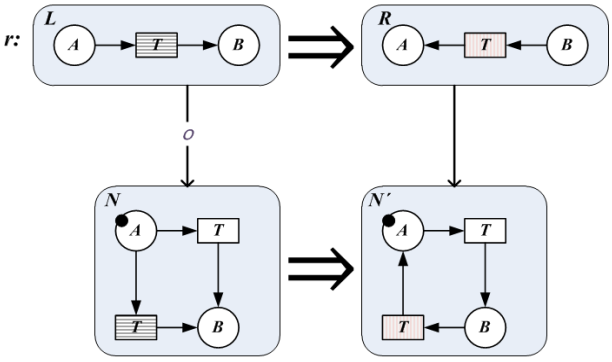

Figure 2. Transformation $N \overset{r}{\Rightarrow} N'$.

the Petri nets, as labels, names, capacities and other decoration of the net [13]. Other control structures may determine the application of rules. They concern the situation that may or may not be given or they concern the order of the rules to be applied. We first investigate control structure for the transformation part, namely net transformation units, rule priorities and application conditions. Next we deal with control structures for Petri nets, as labels, transition priorities and inhibitor arcs.

3.1. Net Transformation Units

Graph transformation units have been introduced to graph grammars as the basic units for graph programming [17]. Control conditions can be given by regular expressions, describing in which order and how often the rules and imported units are to be applied. A large body of results has been developed since then [18]-[21], see also [22]. Net transformation units are the transfer of graph transformation units to reconfigurable Petri nets. The underlying theory concerns high-level replacement (HLR) systems and is an abstract formulation of transformation systems in terms of category theory. In this frame work HLR units [18] that allow structuring the transformations have been given. Here, we instantiate this theory obtaining net transformation units. A net transformation unit is a tuple $NT = (R, nm: R \to Names, C)$ where R is a finite set of rules, and $C \in \mathcal{C}$ is a control expression defined over the set *Names*. Thus, the control expressions are given over the names of rules and are defined recursively: $r \in nm(R)$ implies $r \in \mathcal{C}$, so each rule name is an atomic control expression and denotes the application of the corresponding rule. $C_1 \in \mathcal{C}$ and $C_2 \in \mathcal{C}$ implies both $C_1; C_2 \in \mathcal{C}$ and $C_1 | C_2 \in \mathcal{C}$. The first denoting the sequential composition of both control expression, the second being the choice operator allows the application of either the expression C_1 or C_2. Last but not east, there is the Kleene closure, denoting that an arbitrary number of iterations may be executed, so $C_1 \in \mathcal{C}$ implies $(C_1)^* \in \mathcal{C}$.

An example is transformation unit (R, nm, $r1$; $r2^*$; $r3$) where R and nm are given in **Figure 3**. The control expression $r1$; $r2^*$; $r3$ determines that first rule $r1$ is executed, then rule $r2$ is applied repeatedly and at last $r3$ is applied once. So, this unit only yields Petri nets that are circles.

Priorities of rules can be modeled in transformation units as well. But a priority control condition that gives higher priority exclusively to rules is decidable, whereas one that gives higher priority to some imported transformation unit is no longer decidable (see [19]). So, we define them explicitly in this paper. Given a set of rules R, then a partial order $\leq$ over R defines the priorities of the rules. Then a rule r is applicable if there is an occurrence morphism $o: L \to N$ that satisfies the gluing conditions and there is no rule $r \in R$, so that r is applicable and $r \leq r$.

3.2. Application Conditions

Next, we investigate the use of application conditions. Negative Application Conditions (NAC) for reconfigurable Petri nets have been introduced in [23] and provide the possibility to forbid certain rule applications. These conditions restrict the application of a rule forbidding a certain structure to be present before or after applying a rule in a certain context. Such a constraint influences thus each rule application or transformation and therefore changes significantly the properties of the net transformation system. Rules with NACs as shown in **Figure 4** have an additional set of nets NC_i, denoting the forbidden contexts. Formally, a rule is applicable only if match

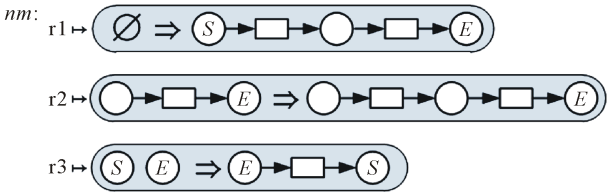

Figure 3. Transformation unit (*R*, *nm*, *r*1; *r*2*; *r*3).

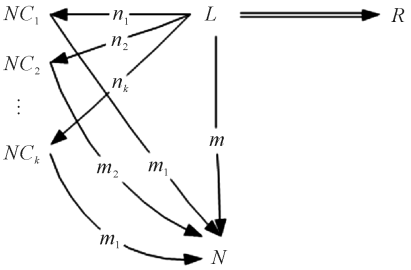

Figure 4. Formal definition of NACs.

m cannot be extended to a match m_i such that $n_i \circ m_i = m$. The nets NC_i together with the morphisms $n_i : L \to NC_i$ are associated to one L, indicating that no extension of m should exist for any $1 \le i \le k$.

An example is given in **Figure 5**, where the rule r deletes no element of L. So, the rule could be applied again. The NAC checks that the net NC is not in the context of the application, thus forbidding the repeated application of the rule r for the same match. So, repeated application of r with different occurrences may yield the net $N1$, but the net $N2$ cannot be obtained by applying r.

A substantial extension of negative application conditions providing a much greater expressiveness are nested application conditions [24]-[26] that have been given in the framework of $\mathcal{M}$-adhesive transformation systems. Stated in terms of reconfigurable Petri nets a nested application ac over a net P and its satisfaction by a morphism $p : P \to N$ is defined inductively:

- $ac = true$ and every morphism satisfies true,
- $ac = \exists(a, ac')$, where $a : P \to C$ is a net morphism and ac' is an application condition over the net C, and a net morphism $p : P \to N$ satisfies ac, if there exists an injective net morphism q with $q \circ a = p$ and q satisfies ac',
- $ac = \neg ac'$, where ac' is a condition over P, and a morphism $p : P \to N$ satisfies ac, if p does not satisfy ac',
- $ac = ac_1 \vee ac_2 \vee \cdots \vee ac_n$, where ac_i belong to some set of conditions over N for $1 \le i \le n$ and a morphism $p : P \to N$ satisfies ac, if p satisfies ac_i for some i.

As an example consider the rule r in **Figure 6**. The application condition ac states that no loops of size 1 or 2 are created by the application of the rule r: This is formalized by $ac : \forall x : \neg\exists(y \vee z)$ where the injective net morphisms in **Figure 6** denote the net expressions x, y and z.

3.3. Further Control Structures

In this section we deal with additional control structures as priorities and labels. Labels and labels that change when a transition fires have been considered for decorated PT nets in [13]. These transition labels may change when the transition fires. This feature is important for the application of a rule after a transition has already fired and cannot be modeled without changing the labels. Considering the tokens in the post place of the transition does not work, because these tokens may be consumed as well. The extension to changing labels is conservative with respect to Petri nets as it does not alter the net's behavior, but it is crucial for the control of rule application and transition firing.

Further important control structures in Petri nets are priorities of transitions and inhibitor arcs (e.g. [27]). The first ensure that an order of transitions restricts the firing, so that a transition may fire only if it has the highest priority of all enabled transitions. The set of transitions T is equipped with a partial order $\le$ on the transitions. A

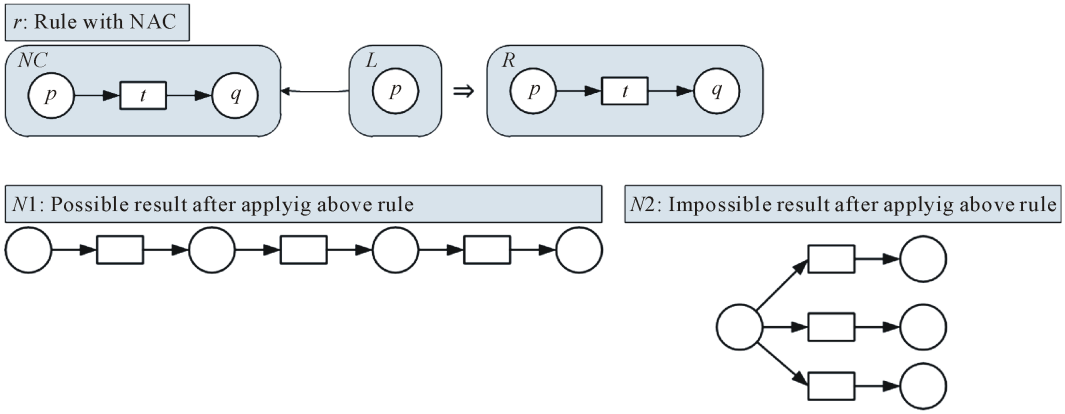

Figure 5. Example for a negative application condition.

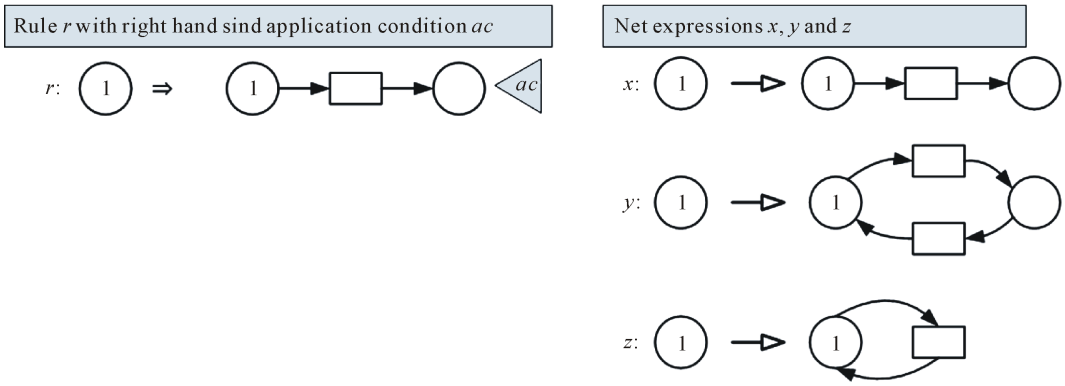

Figure 6. Example of a nested application condition.

transition $t \in T$ is enabled under a marking m, if $pre(t) \le m$, if $cap(t) \ge m + post(t)$ and if for all t being enabled under m we have $t' \le t$. Based on a suitable function this concept has been shown to be compatible with reconfigurable Petri nets in [28]. Inhibitor arc enforce that a transition may only fire when the place connected by an inhibitor arc is empty, thus inhibitor arcs can be generalized by a function $inh: T \to P^{\oplus}$. In [28] this concept has been formulated categorically and is shown to be compatible with reconfigurable Petri nets.

In timed Petri nets (e.g., time Petri nets [29], deterministic timed Petri nets [30] and timed coloured Petri nets [31]) time can be considered a control structure since the firing of transitions is restricted. Based on the concepts of [31] reconfigurable timed Petri nets have been introduced in [32].

4. Summary of Control Structures and Related Work

In the following we summarize the concepts that have been discussed in the previous sections. Moreover, we state for which kind of Petri nets these results have been obtained and point out the corresponding literature. **Table 1** states which concepts affect the application of rules, which affect the firing of transitions, which coordinate both and which concepts enforce an order based on priorities.

Table 2 states if the corresponding control structure is available for the various types of Petri nets and gives the corresponding reference. We have examined place/transition nets, decorated nets and algebraic high-level nets. Since AHOI nets have nets and rules as tokens, they control the dynamics directly and thus have not been considered here.

Table 3 concerns the tool support. Editing and simulating reconfigurable Petri nets requires tool support, there are two tools explicitly dealing with reconfigurable Petri nets ReConNet [8] and the RON-editor [33]. The AGG system [34] is an attributed graph transformation system but has also been used for modeling reconfigurable Petri nets. We have omitted here the large range of other graph transformation tool, that are in principle also capable of modeling reconfigurable Petri nets.

Table 1. Types of control structures.

Control structure	Defined in terms of	Controlling
Negative application conditions	Transformations	Context
Nested application conditions	Transformations	Context
Net transformation units	Order of	Rules
Priorities of rules	Order of	Rules
Labels, names	Nets	Rule application
Changing transition labels	Nets	Coordination of transitions and transformations
Inhibitor arcs	Nets	Firing behavior
Priorities of transitions	Order of	Transition firing
Timed token	Nets	Firing behavior

Table 2. Literature on control structures.

Control structures	Introduced for reconfigurable		
	PT nets	Decorated PT nets	AHL nets
Negative application conditions	[23]	[13]	[16]
Nested application conditions	[25]	-	[2]
Net transformation units	In this paper		-
Priorities of rules	In this paper		-
Labels, names	-	[13]	-
Changing transition labels	-	[13]	-
Inhibitor arcs	-	[28]	-
Priorities of transitions	-	[28]	-
Timed token	[32]	-	-

Table 3. Controls structures in tools.

Control structures	Available in		
	ReConNet	RON	AGG
Negative application conditions	No	Yes	Yes
Nested application conditions	No	No	Yes
Net transformation units	No	No	No
Priorities of rules	No	No	Yes [1]
Labels, names	Yes	Yes	Yes
Changing transition labels	Yes	No	No
Inhibitor arcs	No	No	No
Priorities of transitions	No	No	No
Timed token	No	No	No

5. Conclusions

In the contribution a survey over the control structures for various types of reconfigurable Petri nets has been given. We have investigated several types of reconfigurable Petri nets, namely place/transition nets, decorated nets and algebraic high-level nets. We have given an outline of various control structures for these types reconfigurable Petri nets.

As a result, we can state that there are suitable concepts for control structures for the convenient use of reconfigurable nets. Nevertheless, there only a few of these results are realized in tools. Obviously, future work is to eliminate the open questions, indicated by dashes in **Table 1**, most of which are straightforward. More interesting is the realization of control structures in the tools. Negative application conditions are currently developed for the tool ReConNet.

References

[1] Padberg, J., Hoffmann, K., Ehrig, H., Modica, T., Biermann, E. and Ermel, C. (2007.) Maintaining Consistency in Layered Architectures of Mobile *Ad-Hoc* Networks. In: Dwyer, M.B. and Lopes, A., Eds., *Proceedings of Fundamental Approaches to Software Engineering* 2007, *Vol.* 4422 *of Lecture Notes in Computer Science*, Springer, Berlin, 383-397.

[2] Modica, T. (2012) Formal Modeling, Simulation, and Validation of Communication Platforms. Ph.D. thesis, Technical University of Berlin, Berlin.

[3] Gabriel, K. (2014) Interaction on Human-Centric Communication Platforms: Modelling and Analysis Using Algebraic High-Level Nets and Processes. Ph.D. Thesis, Technische Universität Berlin, Berlin.

[4] Gottmann, S., Nachtigall, N. and Hoffmann, K. (2012) On Modelling Communication in Ubiquitous Computing Systems Using Algebraic Higher Order Nets. *ECEASST*, **51**.

[5] Reiter, F. (2012) Modellierung und Analyse von Szenarien des Living Place mit rekonfigurierbaren Petrinetzen. Bachelor Thesis, Hochschule fr Angewandte Wissenschaften Hamburg.

[6] Llorens, M. and Oliver, J. (2004) Structural and Dynamic Changes in Concurrent Systems: Reconfigurable Petri Nets. *IEEE Transactions on Computers*, **53**, 1147-1158. http://dx.doi.org/10.1109/TC.2004.66

[7] Hoffmann, K., Ehrig, H. and Padberg, J. (2008) Flexible Modeling of Emergency Scenarios Using Reconfigurable Systems. *ECEASST*, **12**.

[8] Ede, M., Hoffmann, K., Oelker, G. and Padberg, J. (2012) ReConNet: A Tool for Modeling and Simulating with Reconfigurable Place/Transition Nets. In: Krause, C. and Westfechtel, B., Eds., *Proceedings of the 7th International Workshop on Graph-Based Tools*, Vol. 54, Electronic Communications of the EASST.

[9] Padberg, J., Ehrig, H. and Ribeiro, L. (1995) Algebraic High-Level Net Transformation Systems. *Mathematical Structures in Computer Science*, **5**, 217-256. http://dx.doi.org/10.1017/S0960129500000724

[10] Gabriel, K. and Ehrig, H. (2012) Modelling Evolution of Communication Platforms and Scenarios Based on Transformations of High-Level Nets and Processes. *Theoretical Computer Science*, **429**, 87-97. http://dx.doi.org/10.1016/j.tcs.2011.12.027

[11] Modica, T. and Hoffmann, K. (2010) Formal Modeling of Communication Platforms Using Reconfigurable Algebraic High-Level Nets. *Electronic Communications of the EASST*, **30**.

[12] Modica, T., Gabriel, K. and Hoffmann, K. (2010) Formalization of Petri Nets with Individual Tokens as Basis for DPO Net Transformations. *Electronic Communications of the EASST*, **40**.

[13] Padberg, J. (2012) Abstract Interleaving Semantics for Reconfigurable Petri Nets. *Electronic Communications of the EASST*, **51**.

[14] Ehrig, H., Golas, U. and Hermann, F. (2010) Categorical Frameworks for Graph Transformation and HLR Systems Based on the DPO Approach. *Bulletin of the EATCS*, **102**, 111-121.

[15] Ehrig, H., Ehrig, K., Prange, U. and Taentzer, G. (2006) Fundamentals of Algebraic Graph Transformation. EATCS Monographs in TCS, Springer, Berlin.

[16] Prange, U. (2008) Towards Algebraic High-Level Systems as Weak Adhesive HLR Categories. *Electronic Notes in Theoretical Computer Science*, **203**, 67-88. http://dx.doi.org/10.1016/j.entcs.2008.10.043

[17] Andries, M., Engels, G., Habel, A., Hoffmann, B., Kreowski, H., Kuske, S., Plump, D., Schürr, A. and Taentzer, G. (1999) Graph Transformation for Specification and Programming. *Science of Computer Programming*, **34**, 1-54. http://dx.doi.org/10.1016/S0167-6423(98)00023-9

[18] Bottoni, P., Hoffmann, K., Parisi-Presicce, F. and Taentzer, G. (2005) High-Level Replacement Units and Their Ter-

mination Properties. *Journal of Visual Languages & Computing*, **16**, 485-507. http://dx.doi.org/10.1016/j.jvlc.2005.07.001

[19] Hölscher, K., Klempien-Hinrichs, R. and Knirsch, P. (2008) Undecidable Control Conditions in Graph Transformation Units. *Electronic Notes in Theoretical Computer Science*, **195**, 95-111. http://dx.doi.org/10.1016/j.entcs.2007.08.028

[20] Kreowski, H., Kuske, S. and von Totth, C. (2010) Stepping from Graph Transformation Units to Model Transformation Units. *Electronic Communications of the EASST*, **30**.

[21] Ermler, M., Kreowski, H., Kuske, S. and von Totth, C. (2011) From Graph Transformation Units via Minisat to Grgen.Net. In: *Applications of Graph Transformations with Industrial Relevance*, 4*th International Symposium*, *AGTIVE* 2011, Budapest, 4-7 October 2011, Revised Selected and Invited Papers, 153-168.

[22] Kreowski, H., Kuske, S. and Rozenberg, G. (2008) Graph Transformation Units—An Overview. In: *Concurrency, Graphs and Models, Essays Dedicated to Ugo Montanari on the Occasion of His* 65*th Birthday*, *Lecture Notes in Computer Science*, Vol. 5065, Springer, Berlin, 57-75.

[23] Rein, A., Prange, U., Lambers, L., Hoffmann, K. and Padberg, J. (2008) Negative Application Conditions for Reconfigurable Place/Transition Systems. In: Ermel, C., de Lara, J. and Heckel, R., Eds., *Proceedings of the Workshop on Graph Transformation and Visual Modeling Techniques* (*GT-VMT*'08), Vol. 10, Electronic Communications of the EASST.

[24] Lambers, L. (2009) Certifying Rule-Based Models Using Graph Transformation. PhD Thesis, Berlin Institute of Technology, Berlin.

[25] Ehrig, H., Golas, U., Habel, A., Lambers, L. and Orejas, F. (2012) $\mathcal{M}$-Adhesive Transformation Systems with Nested Application Conditions. Part 2: Embedding, Critical Pairs and Local Confluence. *Fundamenta Informaticae*, **118**, 35-63.

[26] Ehrig, H., Golas, U., Habel, A., Lambers, L. and Orejas, F. (2014) $\mathcal{M}$-Adhesive Transformation Systems with Nested Application Conditions. Part 1: Parallelism, Concurrency and Amalgamation. *Mathematical Structures in Computer Science*, **24**. http://dx.doi.org/10.1017/S0960129512000357

[27] Chiola, G., Donatelli, S. and Franceschinis, G. (1991) Priorities, Inhibitor Arcs, and Concurrency in P/T Nets. In: *Proceedings of the* 12*th International Conference on Application and Theory of Petri Nets*, Gjern, 182-205.

[28] Padberg, J. (2014) Reconfigurable Decorated PT Nets with Inhibitor Arcs and Transition Priorities. *CoRR*. arXiv:1409.6856.

[29] Berthomieu, B. and Diaz, M. (1991) Modeling and Verification of Time Dependent Systems Using Time Petri Nets. *IEEE Transactions on Software Engineering*, **17**, 259-273. http://dx.doi.org/10.1109/32.75415

[30] Hrúz, B. and Zhou, M.C. (2007) Modeling and Control of Discrete Event Dynamic Systems. Springer, Berlin.

[31] Jensen, K. and Kristensen, L.M. (2009) Coloured Petri Nets—Modelling and Validation of Concurrent Systems. Springer, Berlin.

[32] Gabriel, K., Lingnau, P. and Ermel, C. (2012) Algebraic Approach to Timed Petri Nets. *Electronic Communications of the EASST*, **47**.

[33] Biermann, E., Ermel, C., Hermann, F. and Modica, T. (2007) A Visual Editor for Reconfigurable Object Nets Based on the ECLIPSE Graphical Editor Framework. In: Juhas, G. and Desel, J., Eds., *Proceedings of* 14*th Workshop on Algorithms and Tools for Petri Nets*, *AWPN*'07, Universität Koblenz-Landau, GI Special Interest Group on Petri Nets and Related System Models.

[34] AGG (2014) The Attributed Graph Grammar System.