Overview of Reconfigurable Petri Nets

Julia Padberg[1(✉)] and Laid Kahloul[2]

[1] Hamburg University of Applied Sciences, Hamburg, Germany
julia.padberg@haw-hamburg.de
[2] LINFI Laboratory, Computer Science Department,
Biskra University, Biskra, Algeria
l.kahloul@univ-biskra.dz

Abstract. The evolution in software and hardware systems from classical systems with rigid structures to open, dynamic, and flexible structures has inspired the extension of Petri nets to reconfiguration. The idea of reconfiguring Petri nets was launched in the early nineties and since then has been developed by several researchers at different levels of formalization. Researchers in this field have achieved a large amount of theoretical results and of practical applications. The aim of this paper is to present an overview of reconfigurable Petri nets dealing with several aspects including: the fundamental, theoretical basis, application domains, results at the verification/analysis level as well as practical tools. The paper finally discusses some future research directions.

Keywords: Reconfigurable Petri nets · Petri net transformations
Dynamic infrastructures

1 Introduction

The characteristic feature of reconfigurable Petri nets, consisting of a Petri net and a set of rules that can modify it, is the possibility to discriminate between different levels of change. They provide powerful and intuitive formalisms to model dynamic software or hardware systems that are executed in dynamic infrastructures. These infrastructures are dynamic since they are subject to change as well and since they support various applications that may share some resources. Such dynamic software or hardware systems have become increasingly more common but are difficult to handle. Modelling and simulating dynamic systems require both the representation of their processes and of the system changes within one model. As the underlying type of Petri net can vary (for example being place/transition nets, object nets, timed and/or stochastic nets, or high-level nets) this approach can be considered a family of formal modelling techniques. Reconfigurable Petri nets are an instantiation of abstract transformation systems that are formulated in category theory. The fundamental idea is to characterize those categories that allow double-pushout transformations: therefore only the diagrammatic descriptions are needed. This has the advantage of a thorough theory that yields a vast amount of results concerning the transformation part.

R. Heckel and G. Taentzer (Eds.): Ehrig Festschrift, LNCS 10800, pp. 201–222, 2018.
https://doi.org/10.1007/978-3-319-75396-6_11

Reconfigurable Petri nets have been applied in various application areas where complex coordination and structural adaptation at run-time are required (e.g. mobile ad-hoc networks [62], communication spaces [21,51], ubiquitous computing [10,24], concurrent systems [45], workflows in dynamic infrastructures [29], flexible manufacturing systems [71], reconfigurable manufacturing systems [31]). They improve the expressiveness of Petri nets as they increase flexibility and change while allowing the transitions to fire. This greater expressiveness yields rich models with very large state spaces. The state space of the model is even more complex because states are not capturing only the change of the markings but also structure and connectivity changes [2,57]. The reachability graph is composed of several subgraphs [57], each representing the state space of each accessible configuration in the modelled reconfigurable system. To deal with such growing complexity there are two main ideas: relying on invariant properties or to check the state space for such properties. The first idea is followed mainly in a more informal approach, the second leads to explicit model checking of reconfigurable Petri nets, both are discussed in Sect. 5.

The paper is organized as follows: The next section sketches related work, followed by the section introducing reconfigurable Petri nets. Section 3 gives their formal definition as well as an ongoing example and in Sect. 4 this notion is extended to several different types of Petri nets. Subsequently, we illustrate various application areas and concentrate on modelling reconfigurable manufacturing systems in Sect. 6. Then we discuss two tools that have been developed explicitly as tools for reconfigurable Petri nets (see Sect. 7). Finally we discuss some ideas concerning future work in the conclusion.

This paper overlaps with previously published papers, it presents and structures their results.

2 Related Work

The work in this area started in the beginning of the nineties [20] with transformation of various Petri net types as a refinement concept. In several papers [20,54,60] the use of net transformations in algebraic high-level nets had been investigated and a rule-based refinement concept had been developed that ensured safety and liveness properties under specific conditions (e.g. in [55,61,63]). Moreover, based on a categorical framework, namely abstract Petri nets [56] that comprises various low- and high-level types of Petri nets, the results on horizontal and vertical structuring had been made available for these net types. At the turn of the century the idea of adaptation of dynamic systems became an important research topic. The notion "reconfigurable Petri nets" had been coined at INRIA [3] where the reconfiguration had first been the replacement of places.

Zero-safe nets are place-transition nets [12] that allow the distinction between observable and hidden states. In [11] reconfigurable nets are defined as a special case of zero-safe nets. In this approach the post-domain of a transition is not static, but depends on the colours of the consumed tokens.

In [2] net rewriting systems have introduced rules based on a partial morphism between left-hand and right-hand side of the rule. In [47] marked-controlled net rewriting system have been based on a place/transition nets and a graph rewriting for changing configurations. In [4] open nets (a generalisation of place/transition nets suited to model open systems interacting with the surrounding environment) are equipped with suitable classes of reconfiguration rules whose application preserves the observational semantics of the net. The "nets-within-nets" formalism was introduced by Valk in [74] and was combined with workflow Petri nets [67] to develop an approach for the specification and code generation of dynamically reconfigurable embedded systems in [75].

Another approach, called "improved net rewriting systems" (INRS) [41–43] concentrates on preserving important Petri net properties, namely liveness, boundedness and reversibility. It is based on a set of fixed building blocks and rewrite rules with fixed interfaces for the left-hand and the right-hand side. INRS is the basis for the application we investigate in Sect. 6.1, namely Reconfigurable Manufacturing Systems (RMSs) [39]. RMSs allow changeable structures at runtime for various kinds of industrial production systems. To model explicitly the reconfiguration of RMSs several variants of reconfigurable Petri have been proposed and applied. The different proposals can be classified into three principal classes: graph transformation based approaches, approaches based on rewriting net systems, and finally hybrid approaches. Based on [19] formalisms and methods have been developed to design, simulate and verify RMSs [75]. Reconfigurable Object Nets (RON) [8,68] have been used to propose an approach for the design, simulation and verification of RMSs [31–34].

Besides graph transformation based approaches and the INRSs based approach there are proposals that combine Petri nets with rewriting logics, π-calculus, or algebraic specifications to define reconfigurable models. In [35,36] rewriting logic is combined with a variant of the recursive Petri nets [26] to describe the reconfigurability through the ability to model dynamic creation of threads. In [77] object Petri nets are combined with π-calculus, where object-oriented Petri net are employed to depict the static structure and behaviours of the RMS while the π-calculus is used to describe the dynamic structure of the system. In [15] adaptive Petri nets are proposed to specify self-adaptive systems. The adaptation is achieved by the learning ability of neural networks. In [13] the authors deal with transformations over Petri nets as algebraic specifications, thus they developed a tool to set up a basic set of transformation primitives, including adding/removal of nodes, changing the marking and setting connections.

3 Reconfigurable Petri Nets

We now define place/transition nets formally, to have a basis for the definition of rules and transformations later on. Subsequently, we present an example from dynamic hardware reconfiguration.

3.1 Basic Concepts

We use the algebraic approach to Petri nets, where the pre- and post-domain functions $pre, post : T \to P^{\oplus}$ map the transitions T to a multiset of places $P^{\oplus}$ given by the set of all linear sums over the set P. A marking is given by $m \in P^{\oplus}$ with $m = \sum_{p \in P} k_p \cdot p$. The multiplicity of a single place p is given by $(\sum_{p \in P} k_p \cdot p)_{|p} = k_p$. The $\leq$ operator can be extended to linear sums: For $m_1, m_2 \in P^{\oplus}$ with $m_1 = \sum_{p \in P} k_p \cdot p$ and $m_2 = \sum_{p \in P} l_p \cdot p$ we have $m_1 \leq m_2$ if and only if $k_p \leq l_p$ for all $p \in P$. The operations "+" and "–" can be extended accordingly.

Definition 1 (Place/transition nets). *A (marked place/transition) net is given by $N = (P, T, pre, post, cap, lab_P, lab_T, m)$ where P is a set of places, T is a set of transitions. $pre : T \to P^{\oplus}$ maps a transition to its pre-domain and $post : T \to P^{\oplus}$ maps it to its post-domain. Moreover $cap : P \to \mathbb{N}_+^{\omega}$ assigns to each place a capacity (either a natural number or infinity ω), $lab_P : P \to A_P$ is a label function mapping places to a name space, $lab_T : T \to A_T$ is a label function mapping transitions to a name space and $m \in P^{\oplus}$ is the marking denoted by a multiset of places.*
A transition $t \in T$ is m-enabled for a marking $m \in P^{\oplus}$ if we have $pre(t) \leq m$ and $\forall p \in P : (m + post(t))_{|p} \leq cap(p)$. The follower marking m' is computed by $m' = m - pre(t) + post(t)$ and represents the result of a firing step $m[t > m'$.

Net morphisms are given as a pair of mappings for the places and the transitions preserving the structure, the decoration and the marking. Given two nets N_1 and N_2 as in Definition 1 a net morphism $f : N_1 \to N_2$ is given by $f = (f_P : P_1 \to P_2, f_T : T_1 \to T_2)$, so that $pre_2 \circ f_T = f_P^{\oplus} \circ pre_1$ and $post_2 \circ f_T = f_P^{\oplus} \circ post_1$ and $m_1(p) \leq m_2(f_P(p))$ for all $p \in P_1$. The labels and the capacity need to remain the same when mapping one net to another. Moreover, the morphism f is called strict if both f_P and f_T are injective and $m_1(p) = m_2(f_P(p))$ holds for all $p \in P_1$. A rule in the algebraic transformation approach is given by three nets called left-hand side L, interface K and right-hand side R, respectively, and a span of two strict net morphisms $K \to L$ and $K \to R$. Then an occurrence morphism $o : L \to N$ is required that identifies the relevant parts of the left hand side in the given net N.

A transformation step $N \overset{(r,o)}{\Longrightarrow} M$ via rule r can be constructed in two steps by the commutative squares (1) and (2) in Fig. 1. Given a rule with an occurrence $o : L \to N$ the *gluing condition* has to be satisfied in order to apply a rule at a given occurrence. Its satisfaction requires that the deletion of a place implies the deletion of the adjacent transitions, and that the deleted place's marking does not contain more tokens than the corresponding place in L. In this collection [37] such double-pushout transformations are explained in more detail for attributed graphs.

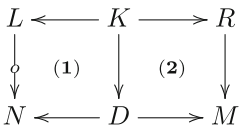


Fig. 1. Net transformation

Reconfigurable place/transition nets exhibit dynamic behaviour using the token game of place/transition nets and using net transformations by applying

rules. So, a reconfigurable net as in Definition 2 combines a net with a set of rules that modify the net [18,19].

Definition 2 (Reconfigurable place/transition nets). *A reconfigurable place/transition net $RN = (N, \mathcal{R})$ is given by a net N and a set of rules $\mathcal{R}$.*

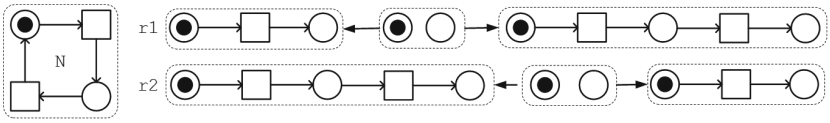


Fig. 2. Cyclic net with rules

Example 1 (Modifying a cyclic process). As an abstract example of a dynamic system we model a cyclic process that can either be executed or modified using the reconfigurable Petri net (N, {r1, r2}). Fig. 2 depicts a simple place/transition net N and the rules r1 and r2. The net describes a cyclic process that executes one step and then returns to the start. The modifications in rule r1 change the process by inserting additional sequential steps. Rule r2 deletes an intermediate step. In Fig. 3 the application of rule r1 to N is given. First a match of the left hand side of the rule is given by the occurrence morphism indicated by the light grey colour of the places and transitions in L and N. The gluing condition holds since the occurrence morphism preserves the token.

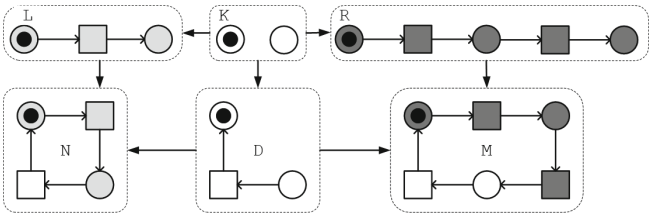


Fig. 3. Application of rule r1 to N

In the first step the transition, which is coloured light grey, is deleted by the construction of the net D and in the second step the intermediate place and its adjacent transitions (coloured dark grey) are added.

3.2 Reconfigurable Computing

Here we give a realistic example that illustrates the use of reconfigurable place/transition nets in dynamic hardware reconfiguration. Reconfigurable computing allows performing several functions on the same hardware with only few modifications. This economic solution avoids the re-fabrication of new hardware when new functions are required. Reconfigurable computing replaces classical fixed digital circuits by FPGA (Field Programmable Gate Array) technologies. An FPGA is a matrix of interconnected logic blocs and it is characterized by its flexibility on both interconnections and logic blocs levels. This flexibility is ensured by programming bits which can be updated rapidly, thus

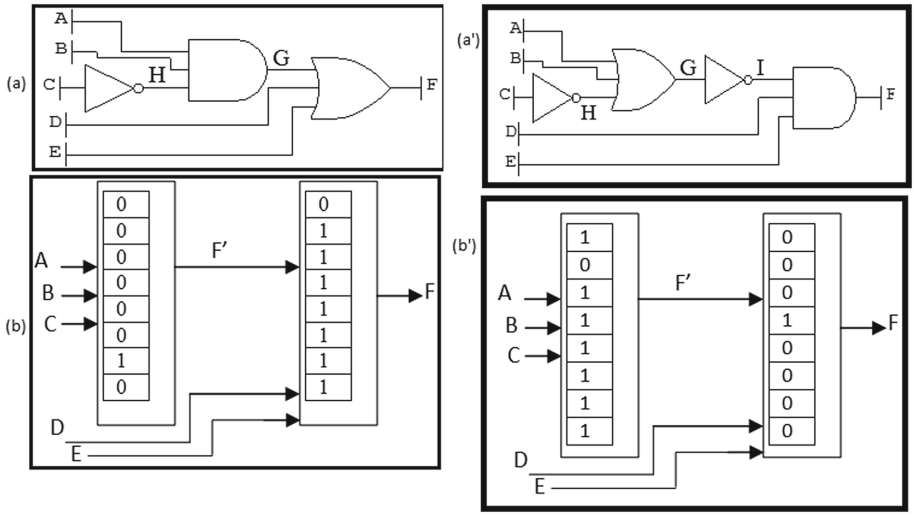


Fig. 4. Reconfigurable computing using LUT

enabling the implementation of several functions in a short time. Usually, the routing mechanism is implemented through programmable gates and the computation exploits lookup-tables (LUTs). A programmable gate is closed or opened due to the value of the programming bit P. A lookup-table is a small circuit which can compute any basic logic function of n inputs by programming its 2^n programming bits. The values of the set of programming bits are saved in an SRAM (Static Random Access Memory). Modifying the values of these programming bits in the SRAM reconfigures the FPGA at two levels: routing connections and computational structure, thus the behaviour of the FPGA is reconfigured. We present in Fig. 4(a) a first configuration which implements the logic function: $F = (A \times B \times \neg C) + D + E$ (known as seat-belt warning light system) as an example. The realisation of this circuit using a LUT table is made using two 3-LUTs (with 8 programming bits) connected as shown in Fig. 4(b). The first LUT uses A, B and C as inputs and generates F'. F', D and E will be the inputs for the second 3-LUT. Figure 4(a′) shows the circuit after a reconfiguration and how this reconfiguration is made through LUT is Fig. 4(b′). In the approach proposed by [38,76] modelling digital circuit (as digital gates) using Petri nets is based on the modelling of signals. The behaviour of a signal x is modelled using two places denoted $x0$ and $x1$ (representing respectively the two signal states 0 and 1) and two transitions $+x$

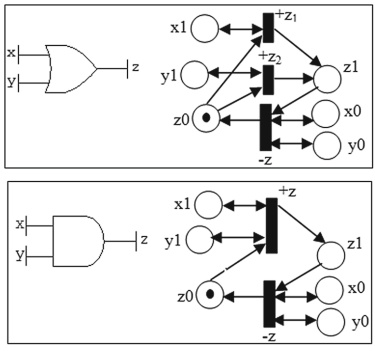


Fig. 5. Petri net models for the "Or gate" and the "And Gate"

and $-x$ (which model respectively the rising of the signal x from 0 to 1 and the falling of the signal x from 1 to 0).

If there are many transitions which rise (resp. fall) the signal x then we denote these transitions as $+x_i$ (resp. $-x_i$). The signal y rises through the transition $+y$ when the signal x is in a 0 level (place $x0$). Then, the signal y will fall by firing the transition $-y$ when the signal x is in the level 1 (place $x1$). The signal x is the input of the gate which will be controlled, eventually, by another circuit.

Using the previous method, the Fig. 5 presents the Petri net models for the AND and OR gates. Hence, we depict in Fig. 6 the models of the two configurations presented in Fig. 4. The formalisation of reconfiguration using the double-pushout (see Fig. 7) requires the definition of a transformation rule from the net TN_1 representing the model of Fig. 6(a) toward the net TN_2 representing the net of Fig. 6(b).

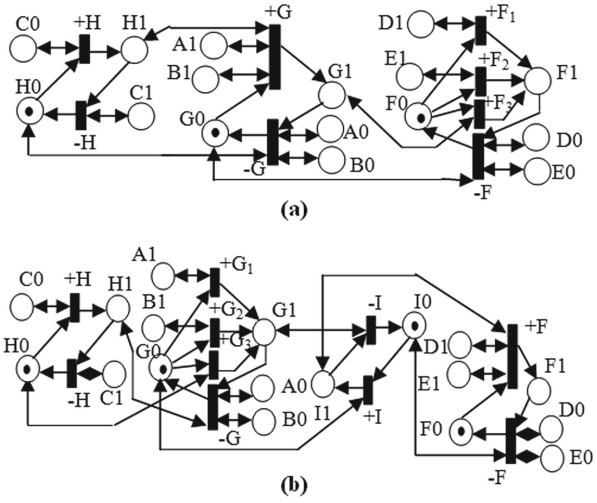


Fig. 6. Net models for the configurations in Fig. 4

4 Types of Reconfigurable Petri Nets

4.1 Reconfigurable Low-level Nets

Place/transition nets as given in Sect. 3 are well-known and widely used. Various types of reconfigurable place/transition nets have been proposed, mostly differing in the additional control structures. In [57] new features have been added to gain an adequate modelling technique where transition labels have been introduced that may change, when the transition is fired. This allows a better coordination of transition firing and rule application, for example one can ensure 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 behaviour, but it is crucial for the coordination of rule application and transition firing.

Reconfigurable place/transition nets with individual tokens [51] have tokens that can be identified as individual objects. Hence, markings are multi-sets of distinguished elements rather than amounts of indistinguishable black tokens. This notion of token facilitates the definition of net processes and hence yields a process semantics [21]. Petri nets do not inherently provide a way to model time but various approaches have been suggested to extend Petri nets by notions of time, as for example [6] or [30]. In [22] timed Petri nets extend place/transition nets attaching time durations to transitions and timestamps to tokens and are

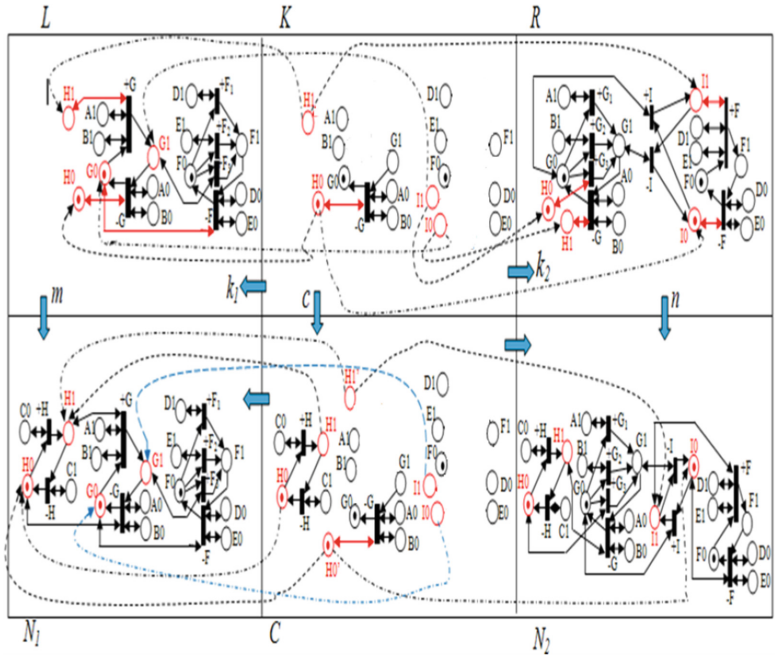


Fig. 7. Double pushout for the example

equipped with rules and transformations. Elementary nets have been shown to be a special case of abstract nets [56]. Hence reconfigurable elementary nets can be considered as well, but they have not be used explicitly.

The above mentioned Petri net types have been proven to yield $\mathcal{M}$-adhesive categories, this means that the results given in Sect. 5.1 are valid for each of these net types.

4.2 Reconfigurable Stochastic Nets

Stochastic Petri nets (SPN) are high level Petri nets that have been proposed to model stochastic and random systems. The most used variant are SPNs in the sense of [48]. They are an extension of timed transition Petri nets where the duration is no longer deterministic but stochastic with a predefined probabilistic law. The Generalised Stochastic Petri Nets (GSPN) [5] are an extension of SPNs where transitions can be of two kinds: stochastic or immediate. The use of GSPNs allows designers to evaluate performance for the modelled systems and to study several quantitative parameters. Introducing reconfiguration into GSPNs is an ambitious issue which will yield a new reconfigurable stochastic Petri net formalism. A first approach has been developed in some recent work [69,70]. There GSPNs have been extended to reconfigurable GSPN using the Improved Net Rewriting Systems [43] to provide a new formalism called INRS-GSPN. This approach has been used to design Stochastic RMSs and to allow performance

evaluation of configurations. Based on the INRS approach the reconfiguration of GSPN is expected to preserve specific required qualities as: liveness, boundedness and reversibility.

4.3 Reconfigurable High-level Nets

Algebraic high-level (AHL) nets are Petri nets combined with algebraic specifications [60]. In contrast to low-level nets AHL nets comprise a data type part, so that the tokens are values in an underlying algebra of the given signature rather than indistinguishable black tokens. The arcs are inscribed by terms over the signature and firing of a transition requires the assignment of these variables to the values of the available tokens. Moreover, transitions are provided with guards, called firing conditions and given by equations. Guards allow the firing of a transition only if the tokens that are to be consumed satisfy the firing conditions of that transition. The operational behaviour of AHL nets is given analogously to the operational behaviour of low-level nets. The activation of a transitions requires an assignment of the variables in the environment of the transition, such that the assigned pre-domain is included in the marking and the firing conditions of the transition hold. This assignment is then used to compute the follower marking, obtained by decreasing the marking by the assigned pre-domain and increasing the result by the assigned post-domain. Algebraic higher-order (AHO) net are high-level nets where tokens can be place/transition nets and net transformation rules. Thus AHO nets follow the paradigm "nets as tokens", introduced by Valk in [73] but extend this paradigm to "nets and rules as tokens". AHO nets are used for controlling firing steps and transformations of low-level nets that has been introduced in [28] with AHL nets that contain place/transition nets and transformation rules as tokens. Reconfigurable Object Nets (RONs) as given in [28] are a restriction of AHO nets, so that firing of RON-transitions may only involve firing of object net transitions, transporting object net tokens through the high-level net, or applying net transformation rules to object nets. Net transformation rules model net modifications such as merging or splitting of object nets, and net refinements. Both AHL nets as well as AHO nets are available with individual tokens [21,51].

The above mentioned high-level net types have also been proven to yield $\mathcal{M}$-adhesive categories, so the results given in Sect. 5.1 hold for each of them.

5 Results

First, we sketch the basics for abstract transformation systems and the results obtained in that way. We now discuss some of the results concerning control structures and verification of reconfigurable nets.

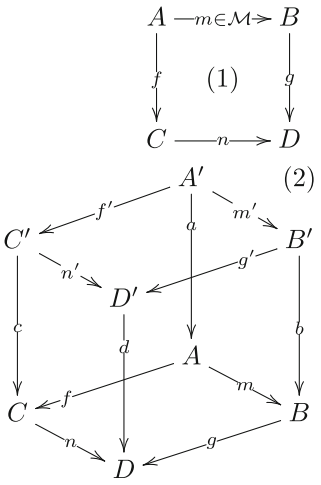
5.1 Results for Abstract Transformation Systems

The basic idea of transforming Petri nets is the same as transforming graphs in the algebraic approach, e.g. in [16]. The theoretical backbone of these

transformations are $\mathcal{M}$-adhesive transformation systems that yield the abstract transformation system. They are formulated in terms of category theory and can be considered as a unifying framework for graph and Petri net transformations providing enough structure that most notions and results from algebraic graph transformation systems hold. Such a categorical approach has the advantage that the results in this framework hold for any category which satisfies the set of assumptions for specific classes of morphisms. $\mathcal{M}$-adhesive transformation systems have been instantiated with various types of graphs, as hypergraphs, attributed and typed graphs, structures, algebraic specifications, various Petri net classes, elementary nets, place/transition nets, Colored Petri nets, or algebraic high-level nets, and more (see [16]). $\mathcal{M}$-adhesive transformation systems allow a uniform description of the different notions and results based on a class $\mathcal{M}$ of specific monomorphisms that have to be PO-PB-compatible, that is:

- Pushouts along $\mathcal{M}$-morphisms exist and $\mathcal{M}$ is stable under pushouts.
- Pullbacks along $\mathcal{M}$-morphisms exist and $\mathcal{M}$ is stable under pullbacks.
- $\mathcal{M}$ contains all identities and is closed under composition.

The fundamental construct for $\mathcal{M}$-adhesive categories and systems are van Kampen squares of $\mathcal{M}$-morphisms.



Definition 3 ($\mathcal{M}$-Adhesive Category). *Given a class M of PO-PB compatible monomorphisms in a category $\mathbf{C}$, then $(\mathbf{C}, \mathcal{M})$ is called $\mathcal{M}$-adhesive category, if pushouts along $\mathcal{M}$-morphisms are $\mathcal{M}$-van Kampen squares, that is for any commutative cube (2) with (1) in the bottom and back faces being pullbacks, the following holds: the top is pushout $\Leftrightarrow$ the front faces are pullbacks.*

An $\mathcal{M}$-adhesive transformation system $AHS = (\mathbf{C}, \mathcal{M}, \mathcal{R})$ consists of an adhesive $\mathcal{M}$-category $(\mathbf{C}, \mathcal{M})$ and a set of rules $\mathcal{R}$.

Based on this categorical framework we have the following results for those Petri net types that have been shown to be adhesive categories:

- Negative application conditions [25] allow specifying undesired context. The rules are equipped with additional nets that show the context in which the rule should not be applied, see Fig. 12 in Sect. 7.2.
- Confluence and independency results as parallel and sequential independence, local Church-Rosser, conflict and causal dependency describe how rules behave in specific contexts. Indepenency conditions are given for two direct transformations being applied to the same net, so that they can be applied in arbitrary order leading to the same result. Properties of dependent transformations have been investigated as well (see e.g. [16]).

- Critical pair analysis as known from term rewriting are used to check for confluence. Critical pairs specify the minimal instance of a conflicting situation. From the set of all critical pairs the items causing conflicts or dependencies are extracted. Local confluence can be shown for abstract transformation systems using the concept of critical pairs (see e.g. [17]).
- Net transformation units have been instantiated from HLR units [9] that are a generalisation of graph transformation units (e.g. [40]). Net transformation units [27] provides syntactic and semantic means for structuring net transformations. They regulate the application of rules to nets encapsulating the rules and control expressions, see also Fig. 12 in Sect. 7.2.

5.2 Control Structures

Control structures to reconfigurable Petri nets are required due to the expressive power of the interplay between rule application and firing behaviour. The available control structures can be differentiated into those that arise from Petri nets, as transition priorities, inhibitor arcs or capacities and those that arise from graph transformation systems as negative application conditions or transformation units.

In [58] priorities for transitions and inhibitor arcs – both well-known concept in Petri nets – have been introduced to reconfigurable Petri nets. The results of $\mathcal{M}$-adhesive transformation system for reconfigurable Petri nets with priorities are ensured by proving the corresponding category to be $\mathcal{M}$-adhesive. Moreover, it was shown that Petri nets with inhibitor arcs yield an $\mathcal{M}$-adhesive category as well.

Other control structures 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. Net transformation units are the transfer of graph transformation units (see [40]) to reconfigurable Petri nets and have been achieved using the abstract formulation of HLR units [9]. Control conditions can be given by regular expressions, describing in which order and how often the rules and imported units are to be applied. For an example see Sect. 7.2.

Negative application conditions for reconfigurable Petri nets have been introduced in [66] 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, again see Sect. 7.2.

5.3 Verification

Ensuring that relevant properties as liveness, reachability of specific states or boundedness hold, is central for the adequate use of a modelling technique. Since the classical analysis techniques fail there are two main possibilities for reconfigurable Petri nets that we discuss subsequently: either these properties are preserved during the transformation or they are proven explicitly.

Invariants. Invariant properties can be achieved if the rules preserve the corresponding properties. Conditions for rules have been formulated so that the rule application ensures safety properties [61] and liveness [72] in the resulting net provided that the original net satisfies these properties.

Although Net Rewriting Systems (NRSs) allow the formalisation of dynamics in Petri nets, they do not preserve properties such as liveness, boundedness (or safeness), and reversibility. Based on the approach of NRSs developed in [2,45], the authors of [42–44] propose Improved Net Rewriting Systems (INRS). The INRS can not only change dynamically the structure of a Petri net but also preserve important behavioural properties. In fact, preserving liveness, boundedness (or safeness), and reversibility is important in several systems such as Reconfigurable Manufacturing Systems (RMSs), where the previous properties are vital to guarantee that the RMS is free from deadlocks, has finite states, and behaves cyclically, respectively. Reconfiguration in INRS replaces some subnet from the source net by another subnet yielding a new net. The approach defines net block class libraries (well-formed net blocks) and the reconfiguration process substitutes a well-formed subnet of any live bounded reversible net with another well-formed net block of the same interface type. The INRS approach was applied in [43] to design reconfigurable Petri net controllers for the supervision of RMSs.

Model Checking. The non-deterministic and concurrent behaviour of reconfigurable Petri nets inhibit the determination of emerging properties. In [64] model checking of reconfigurable place/transition nets has been developed, implemented and proven to be correct. Maude is a mature theory of rewriting logic and is feasible for modelling reconfigurable Petri nets, e.g. [14]. Model-checking of reconfigurable Petri nets [64] is achieved by a conversion of a net and a set of rules into a Maude specification. This specification can be model-checked for properties expressed in linear temporal logic (LTL) using the Maude module LTLR with extensions for rewrite rules and properties such as fairness. The model-checking of reconfigurable nets allows the verification of reachability of states or the absence of deadlocks.

6 Applications

In this section we introduce some of the application areas for reconfigurable Petri nets. First we investigate the use reconfigurable nets for manufacturing systems in some detail. Subsequently, other application areas are merely sketched.

6.1 Reconfigurable Manufacturing Systems

Reconfigurable Manufacturing Systems (RMSs) [39] represent a new innovative approach providing “production systems” with a changeable structure at runtime. Changing the structure can be a solution to satisfy dynamic customers requirement as well as to resolve unpredictable system failures. The use of reconfigurable Petri nets in the design of RMSs offers high level specification, simulation, verification, performance analysis, and code generation at the software

level. Using Reconfigurable Object Nets (RONs) [8] a formal approach [31] for the design, simulation, and verification of RMSs is proposed.

It starts with an informal or semi-formal description of the RMS, builds a RON model for simulation or formal verification. The semi-formal description of RMSs is often given as bloc-diagrams describing tasks and the flow control in the RMS. Figure 8 depicts how the approach is applied by the designer. As a demonstration of the depicted approach in Fig. 8, let's consider the following RMS inspired from [50]. This RMS contains two manufacturing cells (MC_1, MC_2), a storage AS/AR (automated storage and retrieval system), and an AGV (automated guided vehicle). The system requires two raw materials R_1 and R_2 and it produces one final product A (see Fig. 9(a)). The production process passes respectively by MC_1 (Fig. 9(b)) then MC_2 (Fig. 9(c)). The MC_1 is composed of a CNC lathe machine, a CNC milling machine, a robot and a buffer. In MC_1, R_1 and R_2 are processed firstly by the lathe machine before the milling machine. The MC_2 is composed by an assembly machine (which assembles the two products into one product A), a robot and a buffer. During the life cycle of the system, the production process meets

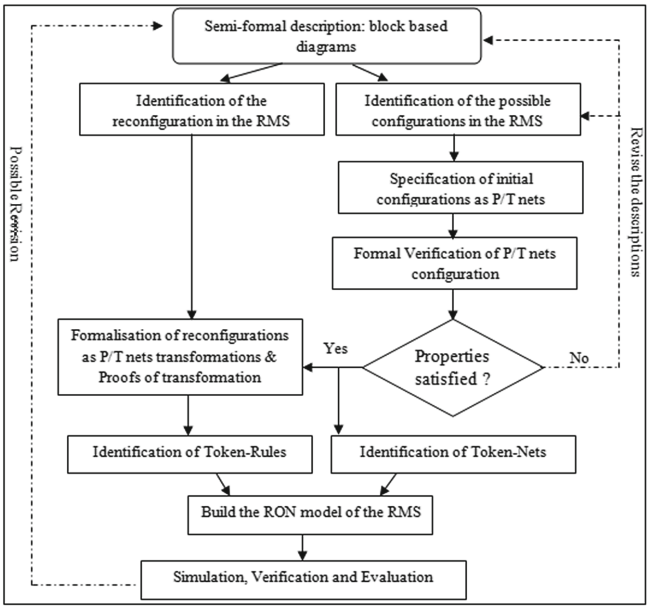


Fig. 8. Designing RMSs using RONs based-approach

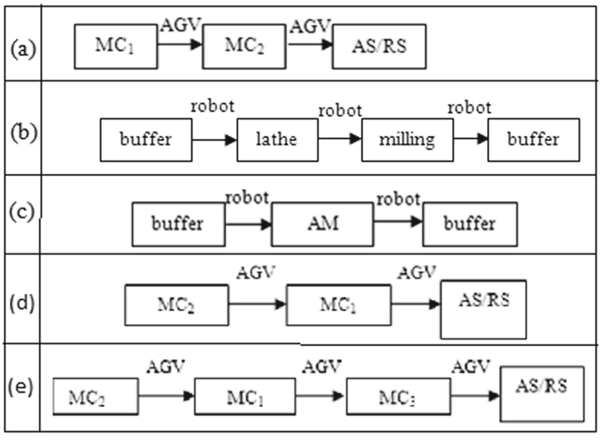


Fig. 9. (a) Manufacturing of product A, (b) Flow in MC_1, (c) Flow in MC_2, (d) Reconfiguration 1: a new product B, (e) Reconfiguration 2: introducing MC_3.

two reconfigurations. The first reconfiguration is triggered by a new customers' requirement for a product B (see Fig. 9(d)). To produce B the flow must be converted (i.e. the assembly is done before the lathe and the milling). The second reconfiguration (see Fig. 9(e)) is triggered by the inspection team of the production process requiring the introduction of a new manufacturing cell MC_3 (inspection cell). The inspection cell MC_3 is composed of a coordinate measuring machine (CMM) and a set of buffers. Using the RON formalism the three configurations are considered as token nets and the reconfigurations are considered as token rules. Figure 10(a) and (b) represent the two first configurations. The reconfiguration is described as a double-pushout (See Fig. 10(c)). As an example, the Fig. 10(d) shows the production (L, I, R) used in the double-pushout.

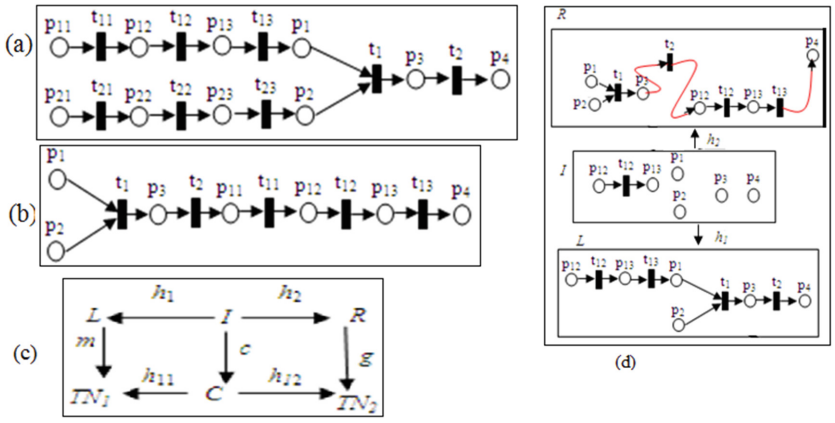


Fig. 10. (a) Initial configuration, (b) Second configuration, (c) Double Pushout, (d) The production rule (L, I, R).

By determining the set of token nets and the set of rules the whole RMS behaviour can be modelled by a RON model as shown in Fig. 11 in Sect. 7.1. This RON model can be used to visualise, simulate, and verify the RMS behaviour.

6.2 Other Applications

Communication Platforms. A general modelling framework for communication platforms and scenarios has been presented in [21] using reconfigurable algebraic high-level 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 [51,52] for a case study on modelling a concrete communication platform – namely Skype. The behaviour of the Skype clients has been modelled 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 (UCSs). Lets computing appear to occur using any device, in any location, and in any format. Underlying technologies comprise the internet, advanced middleware, mobile devices, constantly available networked sensors and microprocessors, and so on. USCs penetrates almost imperceptibly in everyday life. To ensure a solid operation, a UCS needs reliable and efficient communication between its distributed computing components. [24] presents a formal approach based on reconfigurable algebraic higher order nets with individual tokens (AHOI) nets [51]. This approach allows modelling the synchronous and asynchronous communication in UCSs and is used for modelling a smart home. Emergency scenarios using mobile ad-hoc networks have been investigated extensively [10,23,62]. In emergency scenarios, we can obtain an effective coordination among team members constituting a mobile ad-hoc network through the use of net system and rule tokens. From an abstract point of view, mobile ad-hoc networks consist of mobile nodes which communicate with each other independent of a stable infrastructure, while the topology of the network constantly changes depending on the current position of the nodes and their availability. The net structure can be adapted to new requirements of the environment during run time by a set of rules, i.e. token firing and net transformation can be interleaved with each other.

7 Tools

Since the rewriting in reconfigurable Petri nets is in most cases given as a kind of graph transformation, general purpose graph transformation tools as AGG [1] that supports the modelling, the simulation and the analysis of typed attributed graph transformation systems, are likely candidates. But specific tools have been developed as well. In Sects. 7.1 and 7.2 we introduce tools that are in use at this point. The MCReNet-tool [49] is a tool for the modelling and verification of marked-controlled reconfigurable Petri nets [46] and was the first implementation that has explicitly dealt with reconfigurable Petri nets.

7.1 RON-Editor

One of the tools concerned with reconfigurable Petri nets is RON-Editor [7]. The RON-editor is based on reconfigurable object nets [28]. It is an open source and free tool [68]. The RON-editor supports users to create, delete and edit parts of the model like object nets, net transformations rules and a top-level RON. The RON-editor makes several checks (e.g. for correct typing of tokens on RON places, to guarantee that mappings in rules satisfy net morphism properties) that help the user to obtain consistent RONs. Additionally, the editor comprises a simulator using the AGG engine to simulate the application of rules and thus firing of high-level transitions in the RONs created with the editors. The set of visual editors have been realized as Eclipse plug-ins using the Eclipse Modelling Framework (EMF) and Graphical Editor Framework (GEF) plug-ins. As an example, the simulation of the RON model presented in Sect. 6.1 is depicted

in Fig. 11. There the top-right window depicts the system level net (the RON model), the top-left window depicts the object net TN_1 in the place np_1 and the lower window shows the transformation rule (token-rule r_1) which is applied to the object net TN_1. The RON-editor can simulate the behaviour of the system level net as well as the behaviour of the object nets. The transition "transform-transition 1" is green which means that it is enabled.

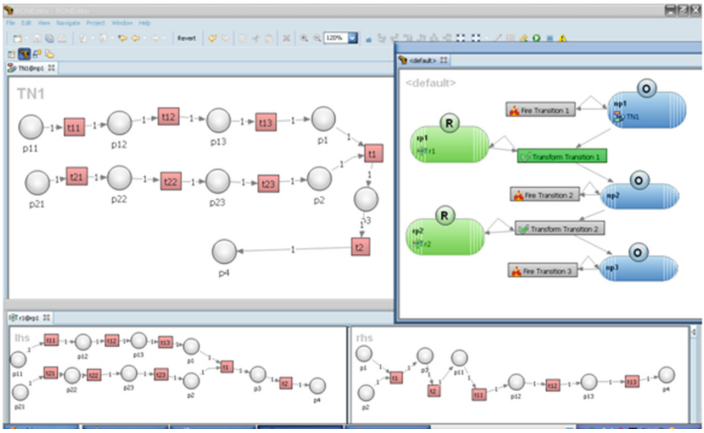


Fig. 11. An example of reconfigurable object nets with RON-Editor. (Color figure online)

7.2 ReConNet

ReConNet [59,65] is an open source project that has been initiated at the HAW Hamburg developing a tool for editing and simulating reconfigurable decorated nets. It provides an intuitive graphic-based user interface that allows the user to create, modify and simulate reconfigurable nets. It facilitates non-deterministic application of rules and firing steps. There are different simulation options executing a definable amount of steps: only transitions are fired, only rules are applied, or both may happen. Control structures that are available comprise negative application conditions, transformation units and dynamic transition labels (see Sect. 5). In Fig. 12 the net `N` from Example 1 in Sect. 3 is depicted together with a third rule `r3` that reverts (various instances and derivations) the arrows of the net. The requirement "the arrows only may be turned if there is no token on the second place" ensures that token do not directly go back. This rule is shown in four windows: the first presents the negative application condition that ensures the requirement. The next three windows present a rule where the intermediate transition is deleted and then a transition with a new label is inserted so that the arrow point in the other direction. Note, that in this illustration the interface's window does not show a net explicitly. The transformation unit's control structure `tu1: (r1 | r2)*;(r3!)` guarantees that `r1` and `r2` (as given in Example 1) are executed arbitrarily often, subsequently they

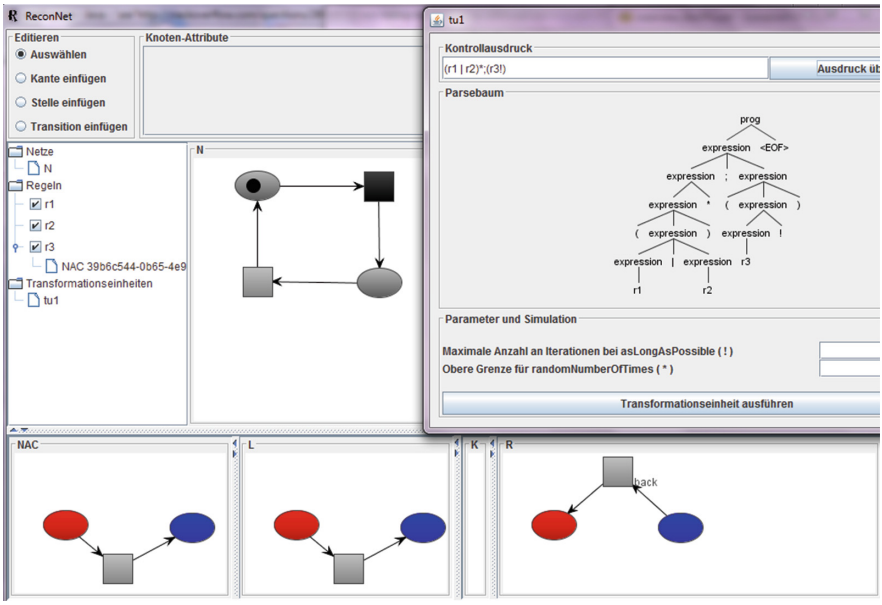


Fig. 12. GUI of ReConNet

are followed by rule `r3`. The exclamation mark `!` denotes that this expression is repeated *as long as possible*. Hence, once the turning has started, it goes on until all arrows have been turned around.

8 Conclusion

In this contribution we have given a comprehensive compilation of the results that have been achieved for various types of reconfigurable Petri nets in the last two decades. Reconfiguration is a topic that is quite virulent in very different areas. For some of these modelling the dynamic change with reconfigurable Petri nets seems to be very promising. We have sketched some application areas and have given examples how to tackle the issue with reconfigurable nets. The theoretical research in reconfigurable Petri nets has provided important results on several types of nets. However, the proposed case studies are often academic ones illustrated only to explain the feasibility of the proposed formal approaches. In order to tackle with these limits, future work has to focus on the following aspects:

– Enlarge the application domain of reconfigurable Petri nets to handle new technologies, for example cloud computing systems and the internet of things. These later are the most suitable systems where mobility, flexibility, and dynamics are inherent characteristics. These systems require new reliable hardware devices and new reliable software protocols and drivers, thus reconfigurable Petri nets should be suggested as a validation and verification technique.

– Invest in the automatic tools for modelling, simulation, and verification of reconfigurable Petri nets. The current tools remain at the prototypic and academic level.
– Extend reconfigurable Petri nets to performance analysis. Most verification results for reconfigurable Petri nets concern qualitative verification. However, in real systems the designer expects often performance evaluation and quantitative measurements. Such analysis is well developed in stochastic Petri nets and stochastic automata with some improved tools like GreatSPIN or UPPAAL. Stochastic graph transformations [4] provide attributed typed graph transformations systems for analysing transformation systems with stochastic methods and is a good basis for Future work to integrate stochastic features to reconfigurable Petri nets more formally.
  This comprises work on the integration of the INRS approach and the abstract transformation systems. We want to achieve the strong theoretical basis of the abstract transformation systems also for the INRS approach. The formulation of building blocks used in INRS independently of the underlying net types (similar to [53]) is the basis for a formal correctness proof.
– Integrate optimisation problems. Another, new application of reconfigurable Petri nets is the optimal configuration in reconfigurable systems. These use often evolutionary algorithms that find the optimal configuration after several reconfigurations of a random initial configuration. Combining reconfigurable Petri nets with evolutionary processing can yield new hybrid methods where both objectives are captured: optimisation and formal verification.

References

1. AGG. http://www.user.tu-berlin.de/o.runge/agg/index.html. Accessed 02 June 2017
2. Badouel, E., Llorens, M., Oliver, J.: Modeling concurrent systems: reconfigurable nets. In: Arabnia, H.R., Mun, Y. (eds.) International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 1568–1574 (2003)
3. Badouel, E., Oliver, J.: Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes within Workflow Systems. Research Report RR-3339. INRIA (1998)
4. Baldan, P., Corradini, A., Ehrig, H., Heckel, R., König, B.: Bisimilarity and behaviour-preserving reconfigurations of open Petri nets. Log. Methods comput. Sci. **4**, 126–142 (2008)
5. Bause, F., Kritzinger, P.S.: Stochastic Petri Nets: An Introduction to the Theory. Vieweg+Teubner Verlag, Cape Town (2002)
6. Berthomieu, B., Diaz, M.: Modeling and verification of time dependent systems using time Petri nets. IEEE Trans. Softw. Eng. **17**(3), 259–273 (1991)
7. Biermann, E., Ermel, C., Hermann, F., Modica, T.: A visual editor for reconfigurable object nets based on the ECLIPSE graphical editor framework. In: 14th Workshop on Algorithms and Tools for Petri Nets (2007)
8. Biermann, E., Modica, T.: Independence analysis of firing and rule-based net transformations in reconfigurable object nets. Electron. Commun. EASST **10**, 1–13 (2008)

9. Bottoni, P., Hoffmann, K., Parisi-Presicce, F., Taentzer, G.: High-level replacement units and their termination properties. J. Vis. Lang. Comput. **16**(6), 485–507 (2005)
10. Bottoni, P., Rosa, F.D., Hoffmann, K., Mecella, M.: Applying algebraic approaches for modeling workflows and their transformations in mobile networks. Mob. Inf. Syst. **2**(1), 51–76 (2006)
11. Bruni, R., Melgratti, H., Montanari, U.: Extending the zero-safe approach to coloured, reconfigurable and dynamic nets. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) ACPN 2003. LNCS, vol. 3098, pp. 291–327. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27755-2_7
12. Bruni, R., Montanari, U.: Transactions and zero-safe nets. In: Ehrig, H., Padberg, J., Juhás, G., Rozenberg, G. (eds.) Unifying Petri Nets. LNCS, vol. 2128, pp. 380–426. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45541-8_12
13. Capra, L.: A pure SPEC-inscribed PN model for reconfigurable systems. In: 2016 13th International Workshop on Discrete Event Systems (WODES), pp. 459–465, May 2016
14. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Quesada, J.F.: Maude: specification and programming in rewriting logic. Theor. Comput. Sci. **285**(2), 187–243 (2002)
15. Ding, Z., Zhou, Y., Zhou, M.: Modeling self-adaptive software systems with learning Petri nets. IEEE Trans. Syst. Man Cybern. Syst. **46**(4), 483–498 (2016)
16. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs in TCS. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-31188-2
17. Ehrig, H., Golas, U., Habel, A., Lambers, L., Orejas, F.: $\mathcal{M}$-adhesive transformation systems with nested application conditions. part 2: embedding, critical pairs and local confluence. Fundam. Inform. **118**(1–2), 35–63 (2012)
18. Ehrig, H., Hoffmann, K., Padberg, J., Prange, U., Ermel, C.: Independence of Net Transformations and Token Firing in Reconfigurable Place/Transition Systems. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 104–123. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73094-1_9
19. Ehrig, H., Padberg, J.: Graph grammars and Petri net transformations. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) ACPN 2003. LNCS, vol. 3098, pp. 496–536. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27755-2_14
20. Ehrig, H., Padberg, J., Ribeiro, L.: Algebraic high-level nets: Petri nets revisited. In: Ehrig, H., Orejas, F. (eds.) ADT/COMPASS -1992. LNCS, vol. 785. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-57867-6
21. Gabriel, K., Ehrig, H.: Modelling of communication platforms using algebraic high-level nets and their processes. In: Heisel, M. (ed.) Software Service and Application Engineering. LNCS, vol. 7365, pp. 10–25. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30835-2_2
22. Gabriel, K., Lingnau, P., Ermel, C.: Algebraic approach to timed Petri nets. Electron. Commun. EASST **47**, 1–14 (2012)
23. Golas, U., Hoffmann, K., Ehrig, H., Rein, A., Padberg, J.: Functorial analysis of algebraic higher-order net systems with applications to mobile ad-hoc networks. ECEASST **40**, 1–24 (2010)
24. Gottmann, S., Nachtigall, N., Hoffmann, K.: On modelling communication in ubiquitous computing systems using algebraic higher order nets. ECEASST **51**, 1–12 (2012)
25. Habel, A., Heckel, R., Taentzer, G.: Graph grammars with negative application conditions. Fundam. Inform. **26**(3/4), 287–313 (1996)

26. Haddad, S., Poitrenaud, D.: Recursive Petri nets. Acta Informatica **44**(7), 463–508 (2007)
27. Hoff, C.: Transformationseinheiten als Kontrollstruktur für rekonfigurierbare Petrinetze in ReConNet. Master's thesis, University of Applied Sciences Hamburg (2016)
28. Hoffmann, K., Ehrig, H., Mossakowski, T.: High-level nets with nets and rules as tokens. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 268–288. Springer, Heidelberg (2005). https://doi.org/10.1007/11494744_16
29. Hoffmann, K., Ehrig, H., Padberg, J.: Flexible modeling of emergency scenarios using reconfigurable systems. ECEASST **12**, 1–20 (2008)
30. Jensen, K., Kristensen, L.M.: Coloured Petri Nets - Modelling and Validation of Concurrent Systems. Springer, Heidelberg (2009). https://doi.org/10.1007/b95112
31. Kahloul, L., Bourekkache, S.: Djouani, K: Designing reconfigurable manufacturing systems using reconfigurable object Petri nets. Int. J. Comput. Integr. Manuf. **29**, 1–18 (2016)
32. Kahloul, L., Bourekkache, S., Djouani, K., Chaoui, A., Kazar, O.: Using high level Petri nets in the modelling, simulation and verification of reconfigurable manufacturing systems. Int. J. Softw. Eng. Knowl. Eng. **24**(03), 419–443 (2014)
33. Kahloul, L., Chaoui, A., Djouani, K., Bourekkache, S., Kazar, O.: Using high level nets for the design of reconfigurable manufacturing systems. In: 1st International Workshop on Petri Nets for Adaptive Discrete-Event Control Systems, pp. 1–19 (2014)
34. Kahloul, L., Djouani, K., Chaoui, A.: Formal study of reconfigurable manufacturing systems: a high level Petri nets based approach. In: Mařík, V., Lastra, J.L.M., Skobelev, P. (eds.) HoloMAS 2013. LNCS (LNAI), vol. 8062, pp. 106–117. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40090-2_10
35. Kheldoun, A., Barkaoui, K., Zhang, J.F., Ioualalen, M.: A high level net for modeling and analysis reconfigurable discrete event control systems. In: Amine, A., Bellatreche, L., Elberrichi, Z., Neuhold, E.J., Wrembel, R. (eds.) CIIA 2015. IAICT, vol. 456, pp. 551–562. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19578-0_45
36. Kheldoun, A., Zhang, J., Barkaoui, K., Ioualalen, M.: A high-level nets based approach for reconfigurations of distributed control systems. In: ADECS Petri Nets, pp. 36–51 (2014)
37. König, B., Nolte, D., Padberg, J., Rensink, A.: A tutorial on graph transformation. In: Festschrift in Memory of Hartmut Ehrig. Springer (2018, accepted)
38. Kondratyev, A., Cortadella, J., Kishinevsky, M., Lavagno, L., Taubin, A.: The use of Petri nets for the design and verification of asynchronous circuits and systems. J. Circuits Syst. Comput. **8**(1), 67–118 (1998)
39. Koren, Y., Shpitalni, M.: Design of reconfigurable manufacturing systems. J. Manuf. Syst. **29**(4), 130–141 (2010)
40. Kreowski, H.-J., Kuske, S., Rozenberg, G.: Graph transformation units – an overview. In: Degano, P., De Nicola, R., Meseguer, J. (eds.) Concurrency, Graphs and Models. LNCS, vol. 5065, pp. 57–75. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68679-8_5
41. Li, J., Dai, X., Meng, Z.: Improved net rewriting systems-based rapid reconfiguration of Petri net logic controllers. In: 31st 2005 Annual Conference of IEEE Industrial Electronics Society, 6 pp. IEEE (2005)
42. Li, J., Dai, X., Meng, Z.: Improved net rewriting system-based approach to model reconfiguration of reconfigurable manufacturing systems. Int. J. Adv. Manuf. Technol. **37**(11–12), 1168–1189 (2008)

43. Li, J., Dai, X., Meng, Z.: Automatic reconfiguration of Petri net controllers for reconfigurable manufacturing systems with an improved net rewriting system-based approach. IEEE Trans. Autom. Sci. Eng. **6**(1), 156–167 (2009)
44. Li, J., Dai, X., Meng, Z., Xu, L.: Improved net rewriting system-extended Petri net supporting dynamic changes. J. Circuits Syst. Comput. **17**(06), 1027–1052 (2008)
45. Llorens, M., Oliver, J.: Structural and dynamic changes in concurrent systems: reconfigurable Petri nets. IEEE Trans. Comput. **53**(9), 1147–1158 (2004)
46. Llorens, M., Oliver, J.: MCReNet: a tool for marked-controlled reconfigurable nets. In: International Conference on Quantitative Evaluation of Systems, pp. 255–256 (2005)
47. Llorens, M., Oliver, J.: A basic tool for the modeling of marked-controlled reconfigurable Petri nets. ECEASST **2**, 1–13 (2006)
48. Marsan, M.A.: Stochastic Petri nets: an elementary introduction. In: Rozenberg, G. (ed.) APN 1988. LNCS, vol. 424, pp. 1–29. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-52494-0_23
49. MCReNet. http://users.dsic.upv.es/~mllorens/MCReNet.htm. Accessed 14 May 2017
50. Meng, X.: Modeling of reconfigurable manufacturing systems based on colored timed object-oriented Petri nets. J. Manuf. Syst. **29**(2–3), 81–90 (2010)
51. Modica, T., Gabriel, K., Hoffmann, K.: Formalization of Petri nets with individual tokens as basis for DPO net transformations. ECEASST **40**, 1–21 (2010)
52. Modica, T., Homann, K.: Formal modeling of communication platforms using reconfigurable algebraic high-level nets. ECEASST **30**, 1–24 (2010)
53. Murata, T.: Petri nets: properties, analysis and applications. Proc. IEEE **77**(4), 541–580 (1989)
54. Padberg, J.: Algebreic high-level net transformation systems: a survey over theory and applications. Bull. EATCS **51**, 102–110 (1993)
55. Padberg, J.: Categorical approach to horizontal structuring and refinement of high-level replacement systems. Appl. Categ. Struct. **7**(4), 371–403 (1999)
56. Padberg, J.: Classification of Petri nets using adjoint functors. In: Salomaa, A., Gheorghe, P., Rozenberg, G. (eds.) Current Trends in Theoretical Computer Science, pp. 171–179. World Scientific, Singapore (2001)
57. Padberg, J.: Abstract interleaving semantics for reconfigurable Petri nets. ECEASST **51**, 1–14 (2012)
58. Padberg, J.: Reconfigurable Petri nets with transition priorities and inhibitor arcs. In: Parisi-Presicce, F., Westfechtel, B. (eds.) ICGT 2015. LNCS, vol. 9151, pp. 104–120. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21145-9_7
59. Padberg, J., Ede, M., Oelker, G., Hoffmann, K.: Reconnet: a tool for modeling and simulating with reconfigurable place/transition nets. ECEASST **54**, 1–11 (2012)
60. Padberg, J., Ehrig, H., Ribeiro, L.: Algebraic high-level net transformation systems. Math. Struct. Comput. Sci. **5**(2), 217–256 (1995)
61. Padberg, J., Gajewsky, M., Ermel, C.: Rule-based refinement of high-level nets preserving safety properties. Sci. Comput. Program. **40**(1), 97–118 (2001)
62. Padberg, J., Hoffmann, K., Ehrig, H., Modica, T., Biermann, E., Ermel, C.: Maintaining consistency in layered architectures of mobile ad-hoc networks. In: Dwyer, M.B., Lopes, A. (eds.) FASE 2007. LNCS, vol. 4422, pp. 383–397. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71289-3_29
63. Padberg, J., Hoffmann, K., Gajewsky, M.: Stepwise introduction and preservation of safety properties in algebraic high-level net systems. In: Maibaum, T. (ed.) FASE 2000. LNCS, vol. 1783, pp. 249–265. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-46428-X_18

64. Padberg, J., Schulz, A.: Model checking reconfigurable Petri nets with maude. In: Echahed, R., Minas, M. (eds.) ICGT 2016. LNCS, vol. 9761, pp. 54–70. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40530-8_4
65. ReConNet. https://reconnetblog.wordpress.com/. Accessed 16 May 2017
66. Rein, A., Prange, U., Lambers, L., Hoffmann, K., Padberg, J.: Negative application conditions for reconfigurable place/transition systems. ECEASST **10**, 1–14 (2008)
67. Richta, T., Janousek, V., Kocí, R.: Petri nets-based development of dynamically reconfigurable embedded systems. PNSE+ ModPE **989**, 203–217 (2013)
68. RON-Editor. http://www.user.tu-berlin.de/o.runge/tfs/projekte/roneditor/. Accessed 24 May 2017
69. Tigane, S., Kahloul, L., Bourekkache, L.: Net rewriting system for GSPN: A RMS case study. In: 2016 International Conference on Advanced Aspects of Software Engineering (ICAASE), pp. 38–45. IEEE (2016)
70. Tigane, S., Kahloul, L., Bourekkache, S.: Reconfigurable stochastic Petri nets for reconfigurable manufacturing systems. In: Borangiu, T., Trentesaux, D., Thomas, A., Leitão, P., Barata Oliveira, J. (eds.) Service Orientation in Holonic and Multi-Agent Manufacturing. SCI, vol. 694, pp. 383–391. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51100-9_34
71. Târnauc, B., Puiu, D., Comnac, V., Suciu, C.: Modelling a flexible manufacturing system using reconfigurable finite capacity Petri nets. In: 13th International Conference on Optimization of Electrical and Electronic Equipment, pp. 1079–1084, May 2012
72. Urbášek, M.: Preserving properties in system redesign: rule-based approach. In: Wirsing, M., Pattinson, D., Hennicker, R. (eds.) WADT 2002. LNCS, vol. 2755, pp. 442–456. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-40020-2_26
73. Valk, R.: Petri nets as token objects. In: Desel, J., Silva, M. (eds.) ICATPN 1998. LNCS, vol. 1420, pp. 1–24. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-69108-1_1
74. Valk, R.: Object Petri nets. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) ACPN 2003. LNCS, vol. 3098, pp. 819–848. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27755-2_23
75. Van Der Aalst, W., Van Hee, K.M.: Workflow Management: Models, Methods, and Systems. MIT press, Cambridge (2004)
76. Yakovlev, A., Koelmans, A., Semenov, A., Kinniment, D.: Modelling, analysis and synthesis of asynchronous control circuits using Petri nets. Integr. VLSI J. **21**(3), 143–170 (1996)
77. Yu, Z., Guo, F., Ouyang, J., Zhou, L.: Object-oriented Petri nets and π-calculus-based modeling and analysis of reconfigurable manufacturing systems. Adv. Mech. Eng. **8**(11), 1–11 (2016). https://doi.org/10.1177/1687814016677698