Independence of Net Transformations and Token Firing in Reconfigurable Place/Transition Systems

Hartmut Ehrig, Kathrin Hoffmann*, Julia Padberg,
Ulrike Prange, and Claudia Ermel

Institute for Software Technology and Theoretical Computer Science
Technical University of Berlin, Germany

Abstract. Reconfigurable place/transition systems are Petri nets with initial markings and a set of rules which allow the modification of the net during runtime in order to adapt the net to new requirements of the environment. In this paper we use transformation rules for place/transition systems in the sense of the double pushout approach for graph transformation. The main problem in this context is to analyze under which conditions net transformations and token firing can be executed in arbitrary order. This problem is solved in the main theorems of this paper. Reconfigurable place/transition systems then are applied in a mobile network scenario.

Keywords: integration of net theory and graph transformations, parallel and sequential independence of net transformations and token firing.

1 Introduction

In [23], the concept of reconfigurable place/transition (P/T) systems has been introduced that is most important to model changes of the net structure while the system is kept running. In detail, a reconfigurable P/T-system consists of a P/T-system and a set of rules, so that not only the follower marking can be computed but also the structure can be changed by rule application to obtain a new P/T-system that is more appropriate with respect to some requirements of the environment. Moreover these activities can be interleaved.

For rule-based transformations of P/T-systems we use the framework of net transformations [17, 18] that is inspired by graph transformation systems [34]. The basic idea behind net transformation is the stepwise development of P/T-systems by given rules. Think of these rules as replacement systems where the left-hand side is replaced by the right-hand side while preserving a context. Petri nets that can be changed, have become a significant topic in the recent years, as the adaption of a system to a changing environment gets more and more important. Application areas cover e.g. computer supported cooperative work,

* This work has been partly funded by the research project forMA$_1$NET (see tfs.cs.tu-berlin.de/formalnet/) of the German Research Council.

J. Kleijn and A. Yakovlev (Eds.): ICATPN 2007, LNCS 4546, pp. 104–123, 2007.

multi agent systems, dynamic process mining or mobile networks. Moreover, this approach increases the expressiveness of Petri nets and allows a formal description of dynamic changes.

In this paper we continue our work by analyzing under which conditions a firing step is independent of a rule-based transformation step. Independence conditions for two firing steps of P/T-systems, i.e. being conflict free, are well-known and closely related to local Church-Rosser properties for graph resp. net transformations (see [34, 17, 18]) that are valid in the case of parallel and sequential independence of rule-based transformations. In [17] conditions for two transformation steps are given in the framework of high-level replacement systems with applications to net transformations, so that these transformation steps applied to the same P/T-system can be executed in arbitrary order, leading to the same result. But up to now it is open under which conditions a net transformation step and a firing step are independent of each other. In more detail, we assume that a given P/T-system represents a certain system state. The next evolution step can be obtained not only by token firing, but also by the application of one of the rules available. Hence, the question arises, whether each of these evolution steps can be postponed after the realization of the other, yielding the same result. Analogously, we ask ourselves if they can be performed in a different order without changing the result.

In Section 2 we present an interesting application of our concept in the area of mobile ad-hoc networks. While Section 3 reviews the notions of reconfigurable nets and net transformations, in Section 4 our main theorems concerning the parallel and sequential independence of net transformation and token firing are achieved. In Section 5 we show how these concepts and results can be put into the more general framework of algebraic higher-order nets. Finally, we outline related work and some interesting aspects of future work in Section 6.

2 Mobile Network Scenario

In this section we will illustrate the main idea of reconfigurable P/T-systems in the area of a mobile scenario. This work is part of a collaboration with some research projects where the main focus is on an adaptive workflow management system for mobile ad-hoc networks, specifically targeted to emergency scenarios[1]. So, as a running example we use a scenario in archaeological disaster/recovery: after an earthquake, a team (led by a team leader) is equipped with mobile devices (laptops and PDAs) and sent to the affected area to evaluate the state of archaeological sites and the state of precarious buildings. The goal is to draw a situation map in order to schedule restructuring jobs. The team is considered as an overall mobile ad-hoc network in which the team leader's device coordinates the other team member devices by providing suitable information (e.g. maps, sensible objects, etc.) and assigning activities. A typical cooperative process to be enacted by a team is shown in Fig. 1 as P/T-system (PN_1, M_1), where we

[1] MOBIDIS - http://www.dis.uniroma1.it/pub/mecella/projects/MobiDIS, MAIS - http://www.mais-project.it, IST FP6 WORKPAD - http://www.workpad-project.eu/

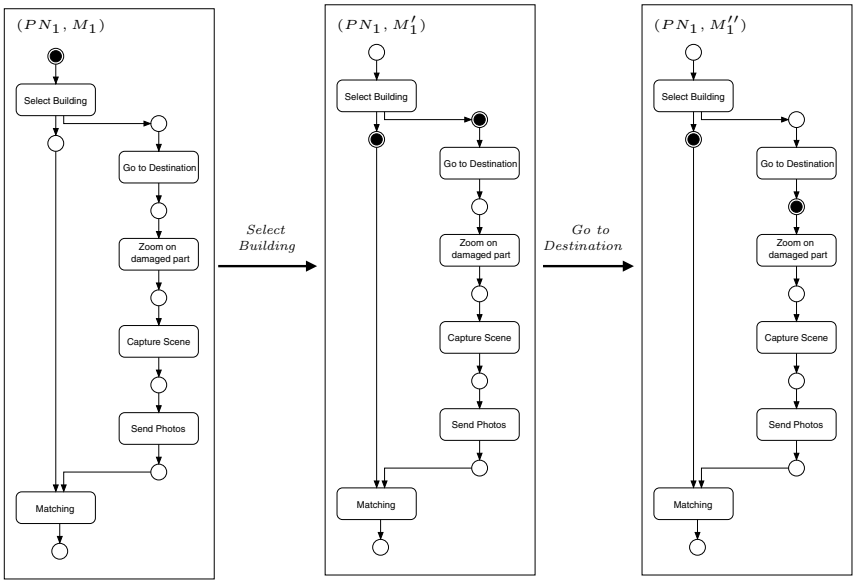


Fig. 1. Firing steps *Select Building* and *Go to Destination*

assume a team consisting of a team leader as picture store device and two team members as camera device and bridge device, respectively.

To start the activities of the camera device the follower marking of the P/T-system (PN_1, M_1) is computed by firing transition *Select Building* and we obtain the new P/T-system (PN_1, M_1') depicted in the middle of Fig. 1. In a next step the task *Go to Destination* can be executed (see right-hand side of Fig. 1).

To predict a situation of disconnection a movement activity of the bridge device has to be introduced in our system. In more detail, the workflow has to be extended by a task to follow the camera device. For this reason we provide the rule $prod_{follow}$ depicted in the upper row in Fig. 2. In general, a rule $prod = ((L, M_L) \xleftarrow{l} (K, M_K) \xrightarrow{r} (R, M_R))$ is given by three P/T-systems called left-hand side, interface, and right-hand side, respectively, and a span of two (specific) P/T-morphisms l and r. For the application of the rule $prod_{follow}$ to the P/T-system (PN_1, M_1) (see Fig. 1) we additionally need a match morphism m that identifies the relevant parts and has to respect the so-called gluing condition (see Section 3). Then the transformation step $(PN_1, M_1) \overset{prod_{follow}}{\Longrightarrow} (PN_2, M_2)$ as shown in Fig. 2 is given as follows: first, the transitions *Go to Destination* and *Send Photos* are deleted and we obtain the intermediate P/T-system (PN_0, M_0); then the transitions *Go to Destination*, *Send Photos* and *Follow Camera Device* together with their (new) environments are added. Note that a positive check of the gluing condition makes sure that the intermediate P/T-system is well-defined. Analogously, the application of the rule $prod_{follow}$ to the

P/T-system (PN_1, M_1') in the middle of Fig. 1 leads to the transformation step $(PN_1, M_1') \overset{prod_{follow}}{\Longrightarrow} (PN_2, M_2')$ in Fig. 3.

Note that in general token game and rule applications cannot be interleaved, e.g. if the transformation rule deletes the transition or a part of the marking used for the token firing. Thus we are looking for conditions such that firing steps and transformation steps can be performed in any order leading to the same P/T-system. In Section 4 we define in more detail conditions to ensure the independence of these activities.

Summarizing, our reconfigurable P/T-system $((PN_1, M_1), \{prod_{follow}\})$ consists of the P/T-system (PN_1, M_1) and the set of rules $\{prod_{follow}\}$ with one rule only. We can consider further rules, e.g. those given in [9,31], leading to a more complex reconfigurable P/T-system. But in this paper we use the simple reconfigurable P/T-system as an example to help the reader understand the main concepts.

3 Reconfigurable P/T-Systems

In this section we formalize reconfigurable P/T-systems. As net formalism we use P/T-systems following the notation of "Petri nets are Monoids" in [28]. In this notation a P/T-net is given by $PN = (P, T, pre, post)$ with pre- and post domain functions $pre, post : T \to P^{\oplus}$ and a P/T-system is given by (PN, M) with marking $M \in P^{\oplus}$, where $P^{\oplus}$ is the free commutative monoid over the set P of places with binary operation $\oplus$, e.g. the monoid notation $M = 2p_1 \oplus 3p_2$ means that we have two tokens on place p_1 and three tokens on p_2. Note that M can also be considered as function $M : P \to \mathbb{N}$ where only for a finite set $P' \subseteq P$ we have $M(p) \geq 1$ with $p \in P'$. We can switch between these notations by defining $\sum_{p \in P} M(p) \cdot p = M \in P^{\oplus}$. Moreover, for $M_1, M_2 \in P^{\oplus}$ we have $M_1 \leq M_2$ if $M_1(p) \leq M_2(p)$ for all $p \in P$. A transition $t \in T$ is M-enabled for a marking $M \in P^{\oplus}$ if we have $pre(t) \leq M$, and in this case the follower marking M' is given by $M' = M \ominus pre(t) \oplus post(t)$ and $(PN, M) \xrightarrow{t} (PN, M')$ is called firing step. Note that the inverse $\ominus$ of $\oplus$ is only defined in $M_1 \ominus M_2$ if we have $M_2 \leq M_1$.

In order to define rules and transformations of P/T-systems we introduce P/T-morphisms which preserve firing steps by Condition (1) below. Additionally they require that the initial marking at corresponding places is increasing (Condition (2)) or even stronger (Condition (3)).

Definition 1 (P/T-Morphisms)
Given P/T-systems $PN_i = (PN_i, M_i)$ with $PN_i = (P_i, T_i, pre_i, post_i)$ for $i = 1, 2$, a P/T-morphism $f : (PN_1, M_1) \to (PN_2, M_2)$ is given by $f = (f_P, f_T)$ with functions $f_P : P_1 \to P_2$ and $f_T : T_1 \to T_2$ satisfying

(1) $f_P^{\oplus} \circ pre_1 = pre_2 \circ f_T$ *and* $f_P^{\oplus} \circ post_1 = post_2 \circ f_T$ *and*
(2) $M_1(p) \leq M_2(f_P(p))$ *for all* $p \in P_1$.

Note that the extension $f_P^{\oplus} : P_1^{\oplus} \to P_2^{\oplus}$ of $f_P : P_1 \to P_2$ is defined by $f_P^{\oplus}(\sum_{i=1}^{n} k_i \cdot p_i) = \sum_{i=1}^{n} k_i \cdot f_P(p_i)$. (1) means that f is compatible with pre- and

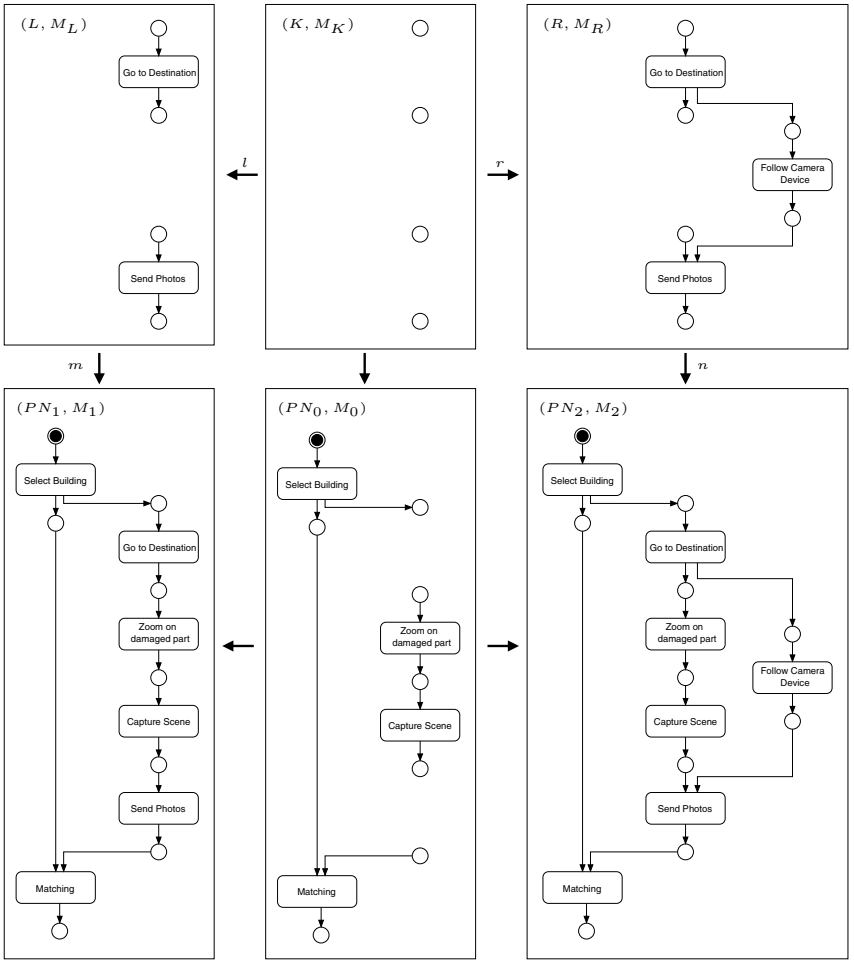


Fig. 2. Transformation step $(PN_1, M_1) \overset{prod_{follow}}{\Longrightarrow} (PN_2, M_2)$

post domain, and (2) that the initial marking of PN_1 at place p is smaller or equal to that of PN_2 at $f_P(p)$.

Moreover the P/T-morphism f is called strict if f_P and f_T are injective and

(3) $M_1(p) = M_2(f_P(p))$ *for all* $p \in P_1$.

The category defined by P/T-systems and P/T-morphisms is denoted by **PTSys** *where the composition of P/T-morphisms is defined componentwise for places and transitions.*

Remark 1. For our morphisms we do not always have $f_P^{\oplus}(M_1) \leq M_2$. E.g. $M_1 = p_1 \oplus p_2, M_2 = p$ and $f_P(p_1) = f_P(p_2) = p$ implies $f_P^{\oplus}(M_1) = 2p > p = M_2$, but $M_1(p_1) = M_1(p_2) = 1 = M_2(p)$.

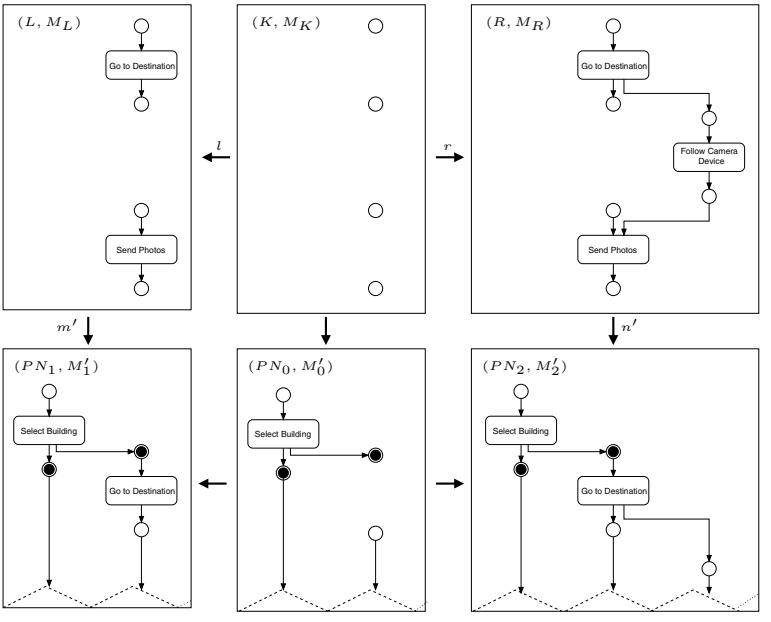


Fig. 3. Transformation step $(PN_1, M_1') \overset{prod_{follow}}{\Longrightarrow} (PN_2, M_2')$

As discussed in our paper [23] we are able to define the gluing of P/T-systems via P/T-morphisms by pushouts in the category **PTSys**. Informally, a pushout in a category **CAT** is a gluing construction of two objects over a specific interface. Especially we are interested in pushouts of the form where l is a strict and c is a general morphism. So, we can apply rules. Vice versa, given the left-hand side of a rule $(K, M_K) \xrightarrow{l} (L, M_L)$ (see Def. 3) and a match m: $(L, M_L) \to (PN_1, M_1)$ we have to construct a P/T-system (PN_0, M_0) such that (1) becomes a pushout. This construction requires the following gluing condition which has to be satisfied in order to apply a rule at a given match. The characterization of specific points is a sufficient condition for the existence and uniqueness of the so called pushout complement (PN_0, M_0), because it allows checking for the applicability of a rule to a given match.

$$\begin{array}{ccc} (K, M_K) & \xrightarrow{\quad l \quad} & (L, M_L) \\ {\scriptstyle c}\downarrow & (1) & \downarrow{\scriptstyle m} \\ (PN_0, M_0) & \longrightarrow & (PN_1, M_1) \end{array}$$

Definition 2 (Gluing Condition for P/T-Systems)
Let $(L, M_L) \xrightarrow{m} (PN_1, M_1)$ be a P/T-morphism and $(K, M_K) \xrightarrow{l} (L, M_L)$ a strict morphism , then the gluing points GP, dangling points DP and the identification points IP of L are defined by

$$\begin{aligned} GP &= l(P_K \cup T_K) \\ DP &= \{p \in P_L | \exists t \in (T_1 \setminus m_T(T_L)) : m_P(p) \in pre_1(t) \oplus post_1(t)\} \\ IP &= \{p \in P_L | \exists p' \in P_L : p \neq p' \wedge m_P(p) = m_P(p')\} \\ &\quad \cup \{t \in T_L | \exists t' \in T_L : t \neq t' \wedge m_T(t) = m_T(t')\} \end{aligned}$$

A P/T-morphism $(L, M_L) \xrightarrow{m} (PN_1, M_1)$ *and a strict morphism* $(K, M_K) \xrightarrow{l} (L, M_L)$ *satisfy the gluing condition, if all dangling and identification points are gluing points, i.e* $DP \cup IP \subseteq GP$, *and* m *is strict on places to be deleted, i.e.*

$$\forall p \in P_L \setminus l(P_K) : M_L(p) = M_1(m(p)).$$

Example 1. In Section 2 examples of P/T-morphisms are given in Fig. 2 by $(K, M_K) \xrightarrow{l} (L, M_L)$ and $(L, M_L) \xrightarrow{m} (PN_1, M_1)$. For the dangling points we have $DP = P_L$ while the set of identification points IP is empty. So, these P/T-morphisms satisfy the gluing condition because the gluing points GP are also equal to the set of places P_L and all places are preserved.

Next we present rule-based transformations of P/T-systems following the double-pushout (DPO) approach of graph transformations in the sense of [34,17], which is restrictive concerning the treatment of unmatched transitions at places which should be deleted. Here the gluing condition forbids the application of rules in this case. Furthermore, items which are identified by a non injective match are both deleted or preserved by rule applications.

Definition 3 (P/T-System Rule)
A rule $prod = ((L, M_L) \xleftarrow{l} (K, M_K) \xrightarrow{r} (R, M_R))$ *of P/T-systems consists of P/T-systems* (L, M_L), (K, M_K), *and* (R, M_R), *called left-hand side (LHS), interface, and right-hand side (RHS) of prod respectively, and two strict P/T-morphisms* $(K, M_K) \xrightarrow{l} (L, M_L)$ *and* $(K, M_K) \xrightarrow{r} (R, M_R)$.

Note that we have not yet considered the firing of the rule nets (L, M_L), (K, M_K) and (R, M_R) as up to now no relevant use could be found. Nevertheless, from a theoretical point of view simultaneous firing of the nets (L, M_L), (K, M_K) and (R, M_R) is easy as the morphisms are marking strict. The firing of only one of these nets would require interesting extensions of the gluing condition.

Definition 4 (Applicability of Rules)
A rule $prod = ((L, M_L) \xleftarrow{l} (K, M_K) \xrightarrow{r} (R, M_R))$ *is called applicable at the match* $(L, M_L) \xrightarrow{m} (PN_1, M_1)$ *if the gluing condition is satisfied for* l *and* m. *In this case we obtain a P/T-system* (PN_0, M_0) *leading to a net transformation step* $(PN_1, M_1) \overset{prod,m}{\Longrightarrow} (PN_2, M_2)$ *consisting of the following pushout diagrams (1) and (2). The P/T-morphism* $n : (R, M_R) \to (PN_2, M_2)$ *is called comatch of the transformation step.*

$$\begin{array}{ccccc}
(L, M_L) & \xleftarrow{\ l\ } & (K, M_K) & \xrightarrow{\ r\ } & (R, M_R) \\
{\scriptstyle m}\downarrow & (\mathbf{1}) & \downarrow{\scriptstyle c} & (\mathbf{2}) & \downarrow{\scriptstyle n} \\
(PN_1, M_1) & \xleftarrow[\ l^*\ ]{} & (PN_0, M_0) & \xrightarrow[\ r^*\ ]{} & (PN_2, M_2)
\end{array}$$

Now we are able to define reconfigurable P/T-systems, which allow modifying the net structure using rules and net transformations of P/T-systems.

Definition 5 (Reconfigurable P/T-Systems)
Given a P/T-system (PN, M) and a set of rules $RULES$, a reconfigurable P/T-system is defined by $((PN, M), RULES)$.

Examples of rule applications and of a reconfigurable P/T-system can be found in Section 2.

4 Independence of Net Transformations and Token Firing

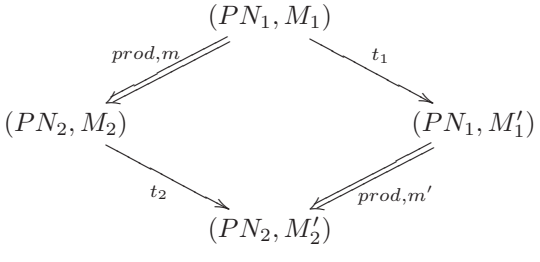
In this section we analyze under which conditions net transformations and token firing of a reconfigurable P/T-system as introduced in Section 3 can be executed in arbitrary order. These conditions are called (co-)parallel and sequential independence. Note that independence conditions for two firing steps of P/T-systems are well-known and independence of two transformation steps is analyzed already for high-level replacement systems with applications to Petri net transformations in [17]. We start with the situation where a transformation step and a firing step are applied to the same P/T-system. This leads to the notion of parallel independence.

Definition 6 (Parallel Independence)
A transformation step $(PN_1, M_1) \overset{prod,m}{\Longrightarrow} (PN_2, M_2)$ of P/T-systems and a firing step $(PN_1, M_1) \xrightarrow{t_1} (PN_1, M_1')$ for $t_1 \in T_1$ are called parallel independent if

(1) *t_1 is not deleted by the transformation step and*
(2) *$M_L(p) \leq M_1'(m(p))$ for all $p \in P_L$ with $(L, M_L) = LHS(prod)$.*

Parallel independence allows the execution of the transformation step and the firing step in arbitrary order leading to the same P/T-system.

Theorem 1 (Parallel Independence). *Given parallel independent steps $(PN_1, M_1) \overset{prod,m}{\Longrightarrow} (PN_2, M_2)$ and $(PN_1, M_1) \xrightarrow{t_1} (PN_1, M_1')$ with $t_1 \in T_1$ then there is a corresponding $t_2 \in T_2$ with firing step $(PN_2, M_2) \xrightarrow{t_2} (PN_2, M_2')$ and a transformation step $(PN_1, M_1') \overset{prod,m'}{\Longrightarrow} (PN_2, M_2')$ with the same marking M_2'.*



Remark 2. Cond. (1) in Def. 6 is needed to fire t_2 in (PN_2, M_2), and Cond. (2) in Def. 6 is needed to have a valid match m' in (PN_1, M_1'). Note that $m'(x) = m(x)$ for all $x \in P_L \cup T_L$.

Proof. Parallel independence implies that $t_1 \in T_1$ is preserved by the transformation step $(PN_1, M_1) \overset{prod,m}{\Longrightarrow} (PN_2, M_2)$. Hence there is a unique $t_0 \in T_0$ with $l^*(t_0) = t_1$. Let $t_2 = r^*(t_0) \in T_2$ in the following pushouts (1) and (2), where l^* and r^* are strict.

$$\begin{array}{ccccc} (L, M_L) & \xleftarrow{l} & (K, M_K) & \xrightarrow{r} & (R, M_R) \\ {\scriptstyle m}\downarrow & (\mathbf{1}) & \downarrow & (\mathbf{2}) & \downarrow{\scriptstyle n} \\ (PN_1, M_1) & \xleftarrow[l^*]{} & (PN_0, M_0) & \xrightarrow[r^*]{} & (PN_2, M_2) \end{array}$$

Now t_1 being enabled under M_1 in PN_1 implies $pre_1(t_1) \leq M_1$. Moreover, l^* and r^* strict implies $pre_0(t_0) \leq M_0$ and $pre_2(t_2) \leq M_2$. Hence t_2 is enabled under M_2 in PN_2 and we define $M_2' = M_2 \ominus pre_2(t_2) \oplus post_2(t_2)$.

Now we consider the second transformation step, with m' defined by $m'(x) = m(x)$ for $x \in P_L \cup T_L$.

$$\begin{array}{ccccc} (L, M_L) & \xleftarrow{l} & (K, M_K) & \xrightarrow{r} & (R, M_R) \\ {\scriptstyle m'}\downarrow & (\mathbf{1}') & \downarrow & (\mathbf{2}') & \downarrow{\scriptstyle n'} \\ (PN_1, M_1') & \xleftarrow[l^{*\prime}]{} & (PN_0, M_0') & \xrightarrow[r^{*\prime}]{} & (PN_2, M_2'') \end{array}$$

m' is a P/T-morphism if for all $p \in P_L$ we have

(*a*) $M_L(p) \leq M_1'(m'(p))$,

and the match m' is applicable at M_1', if

(*b*) $IP \cup DP \subseteq GP$ and for all $p \in P_L \setminus l(P_K)$ we have $M_L(p) = M_1'(m(p))$ (see gluing condition in Def. 2).

Cond. (*a*) is given by Cond. (2) in Def. 6, because we assume that $(PN_1, M_1) \overset{prod,m}{\Longrightarrow} (PN_2, M_2)$ and $(PN_1, M_1) \xrightarrow{t_1} (PN_1, M_1')$ with $t_1 \in T_1$ are parallel independent. Moreover, the match m being applicable at M_1 implies $IP \cup DP \subseteq GP$ and for all $p \in P_L \setminus l(P_K)$ we have $M_L(p) = M_1(m(p)) = M_1'(m(p))$ by Lemma 1 below using the fact that there is a firing step $(PN_1, M_1) \xrightarrow{t_1} (PN_1, M_1')$. The application of *prod* along m' leads to the P/T-system (PN_2, M_2''), where $l^{*\prime}(x) = l^*(x)$, $r^{*\prime}(x) = r^*(x)$ for all $x \in P_0 \cup T_0$, and $n^{*\prime}(x) = n^*(x)$ for all $x \in P_R \cup T_R$.

Finally, it remains to show that $M_2' = M_2''$. By construction of the transformation steps $(PN_1, M_1) \overset{prod,m}{\Longrightarrow} (PN_2, M_2)$ and $(PN_1, M_1') \overset{prod,m'}{\Longrightarrow} (PN_2, M_2'')$ we have

(1) for all $p_0 \in P_0$: $M_2(r^*(p_0)) = M_0(p_0) = M_1(l^*(p_0))$,
(2) for all $p \in P_R \setminus r(P_K)$: $M_2(n(p)) = M_R(p)$,
(3) for all $p_0 \in P_0$: $M_2''(r^*(p_0)) = M_0'(p_0) = M_1'(l^*(p_0))$ and
(4) for all $p \in P_R \setminus r(P_K)$: $M_2''(n'(p)) = M_R(p)$.

By construction of the firing steps $(PN_1, M_1) \xrightarrow{t_1} (PN_1, M_1')$ and $(PN_2, M_2) \xrightarrow{t_2} (PN_2, M_2')$ we have

(5) for all $p_1 \in P_1$: $M_1'(p_1) = M_1(p_1) \ominus pre_1(t_1)(p_1) \oplus post_1(t_1)(p_1)$ and
(6) for all $p_2 \in P_2$: $M_2'(p_2) = M_2(p_2) \ominus pre_2(t_2)(p_2) \oplus post_2(t_2)(p_2)$.

Moreover, l^* and r^* strict implies the injectivity of l^* and r^* and we have

(7) for all $p_0 \in P_0$: $pre_0(t_0)(p_0) = pre_1(t_1)(l^*(p_0)) = pre_2(t_2)(r^*(p_0))$ and
$post_0(t_0)(p_0) = post_1(t_1)(l^*(p_0)) = post_2(t_2)(r^*(p_0))$.

To show that this implies

(8) $M_2' = M_2''$,

it is sufficient to show

(8a) for all $p \in P_R \setminus r(P_K)$: $M_2''(n'(p)) = M_2'(n(p))$ and
(8b) for all $p_0 \in P_0$: $M_2''(r^*(p_0)) = M_2'(r^*(p_0))$.

First we show that condition (8a) is satisfied. For all $p \in P_R \setminus r(P_K)$ we have

$$M_2''(n'(p)) \overset{(4)}{=} M_R(p) \overset{(2)}{=} M_2(n(p)) \overset{(6)}{=} M_2'(n(p))$$

because $n(p)$ is neither in the pre domain nor in the post domain of t_2, which are in $r^*(P_0)$ because t_2 is not created by the rule (see Lemma 1, applied to the inverse rule $prod^{-1}$).

Next we show that condition (8b) is satisfied. For all $p_0 \in P_0$ we have

$$\begin{aligned}
M_2''(r^*(p_0)) &\overset{(3)}{=} M_0'(p_0) \\
&\overset{(3)}{=} M_1'(l^*(p_0)) \\
&\overset{(5)}{=} M_1(l^*(p_0)) \ominus pre_1(t_1)(l^*(p_0)) \oplus post_1(t_1)(l^*(p_0)) \\
&\overset{(1) \text{ and } (7)}{=} M_2(r^*(p_0)) \ominus pre_2(t_2)(r^*(p_0)) \oplus post_2(t_2)(r^*(p_0)) \\
&\overset{(6)}{=} M_2'(r^*(p_0))
\end{aligned}$$

It remains to show Lemma 1 which is used in the proof of Theorem 1.

Lemma 1. *For all $p \in P_L \setminus l(P_K)$ we have $m(p) \notin dom(t_1)$, where $dom(t_1)$ is union of pre- and post domain of t_1, and t_1 is not deleted.*

Proof. Assume $m(p) \in dom(t_1)$.

Case 1 ($t_1 = m(t)$ for $t \in T_L$): t_1 not being deleted implies $t \in l(T_K)$. Hence there exists $p' \in dom(t) \subseteq l(P_K)$, such that $m(p') = m(p)$; but this is a contradiction to $p \in P_L \setminus l(P_K)$ and the fact that m cannot identify elements of $l(P_K)$ and $P_L \setminus l(P_K)$.

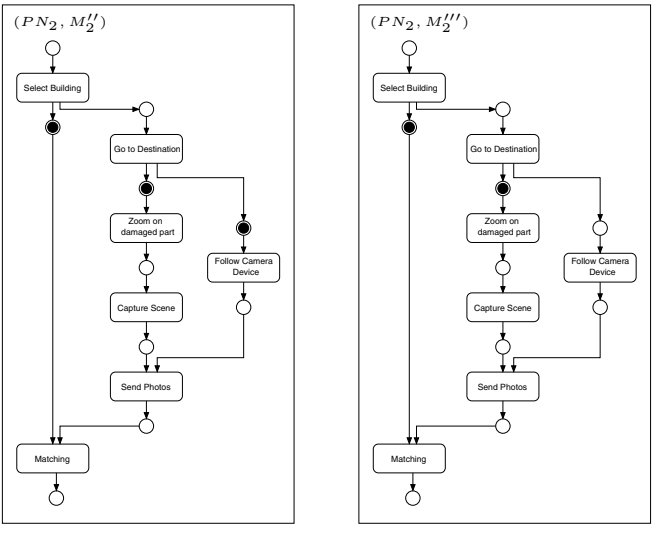


Fig. 4. P/T-systems (PN_2, M_2'') and (PN_2, M_2''')

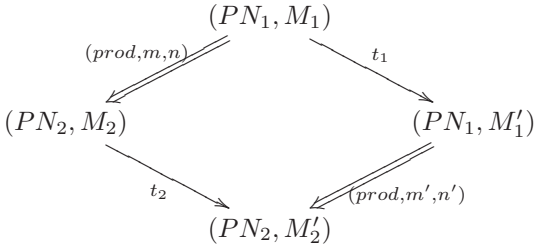
Case 2 ($t_1 \notin m(T_L)$): $m(p) \in dom(t_1)$ implies by the gluing condition in Def. 2, that $p \in l(P_K)$, but this is a contradiction to $p \in P_L \setminus l(P_K)$.

Example 2. The firing step $(PN_1, M_1) \overset{Select\ Building}{\longrightarrow} (PN_1, M_1')$ (see Fig. 1) and the transformation step $(PN_1, M_1) \overset{prod_{follow}}{\Longrightarrow} (PN_2, M_2)$ (see Fig. 2) are parallel independent because the transition *Select Building* is not deleted by the transformation step and the marking M_L is empty. Thus, the firing step can be postponed after the transformation step or, vice versa, the rule $prod_{follow}$ can be applied after token firing yielding the same result (PN_2, M_2') in Fig. 5.

In contrast the firing step $(PN_1, M_1') \overset{Go\ to\ Destination}{\longrightarrow} (PN_1, M_1'')$ (see Fig. 1) and the transformation step $(PN_1, M_1') \overset{prod_{follow}}{\Longrightarrow} (PN_2, M_2')$ (see Fig. 3) are not parallel independent because the transition *Go to Destination* is deleted and afterwards reconstructed by the transformation step (it is not included in the interface K). In fact, the new transition *Go to Destination* in (PN_2, M_2') could be fired leading to (PN_2, M_2'') (see Fig. 4) and vice versa we could fire *Go to Destination* in (PN_1, M_1') and then apply $prod_{follow}$ leading to (PN_2, M_2''') (see Fig. 4), but we would have $M_2'' \neq M_2'''$.

In the first diagram in Theorem 1 we have required that the upper pair of steps is parallel independent leading to the lower pair of steps. Now we consider the situations that the left, right or lower pair of steps are given - with a suitable notion of independence - such that the right, left and upper part of steps can be constructed, respectively.

Definition 7 (Sequential and Coparallel Independence). *In the following diagram with $LHS(prod) = (L, M_L)$, $RHS(prod) = (R, M_R)$, m and m' are matches and n and n' are comatches of the transformation steps with $m(x) = m'(x)$ for $x \in P_L \cup T_L$ and $n(x) = n'(x)$ for $x \in P_R \cup T_R$, we say that*



1. *the left pair of steps, short $((prod, m, n), t_2)$, is sequentially independent if*
   (a) *t_2 is not created by the transformation step*
   (b) *$M_R(p) \leq M_2'(n(p))$ for all $p \in P_R$*
2. *the right pair of steps, short $(t_1, (prod, m', n'))$, is sequentially independent if*
   (a) *t_1 is not deleted by the transformation step*
   (b) *$M_L(p) \leq M_1(m'(p))$ for all $p \in P_L$*
3. *the lower pair of steps, short $(t_2, (prod, m', n'))$, is coparallel independent if*
   (a) *t_2 is not created by the transformation step*
   (b) *$M_R(p) \leq M_2(n'(p))$ for all $p \in P_R$*

Example 3. The pair of steps (*Select Building*, $(prod_{follow}, m', n')$) depicted in Fig. 5 is sequentially independent because the transition *Select Building* is not deleted by the transformation step and the marking M_L is empty. Analogously, the pair of steps $((prod_{follow}, m, n)$, *Select Building*) depicted in Fig. 6 is sequentially independent because the transition *Select Building* is not created by the transformation step and the marking M_R is empty. For the same reason the pair (*Select Building*,$(prod_{follow}, m', n')$) is coparallel independent.

Remark 3. Note that for $prod = ((L, M_L) \stackrel{l}{\leftarrow} (K, M_K) \stackrel{r}{\rightarrow} (R, M_R))$ we have $prod^{-1} = ((R, M_R) \stackrel{r}{\leftarrow} (K, M_K) \stackrel{l}{\rightarrow} (L, M_L))$ and each direct transformation $(PN_1, M_1) \stackrel{prod}{\Longrightarrow} (PN_2, M_2)$ with match m, comatch n and pushout diagrams (1) and (2) as given in Def. 4 leads to a direct transformation $(PN_2, M_2) \stackrel{prod^{-1}}{\Longrightarrow} (PN_1, M_1)$ with match n and comatch m by interchanging pushout diagrams (1) and (2).

Given a firing step $(PN_1, M_1) \stackrel{t_1}{\longrightarrow} (PN_1, M_1')$ with $M_1' = M_1 \ominus pre_1(t_1) \oplus post_1(t_1)$ we can formally define an inverse firing step $(PN_1, M_1') \stackrel{t_1^{-1}}{\longrightarrow} (PN_1, M_1)$ with $M_1 = M_1' \ominus post_1(t_1) \oplus pre_1(t_1)$ if $post_1(t_1) \leq M_1'$, such that firing and inverse firing are inverse to each other.

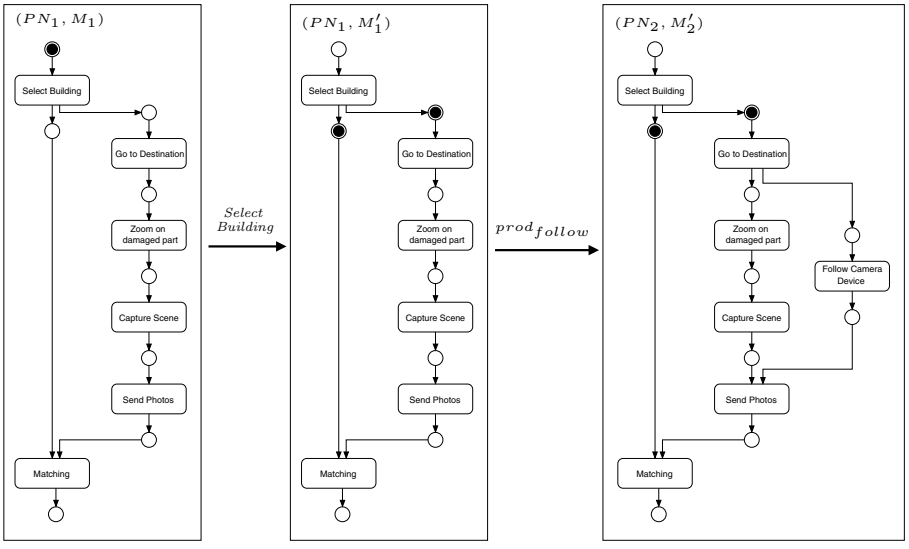
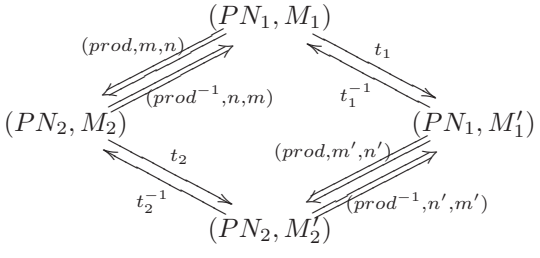


Fig. 5. Pair of steps (*Select Building*, $(prod_{follow}, m', n')$)

Formally all the notions of independence in Def. 7 can be traced back to parallel independence using inverse transformation steps based on $(prod^{-1}, n, m)$ and $(prod^{-1}, n', m')$ and inverse firing steps t_1^{-1} and t_2^{-1} in the following diagram.

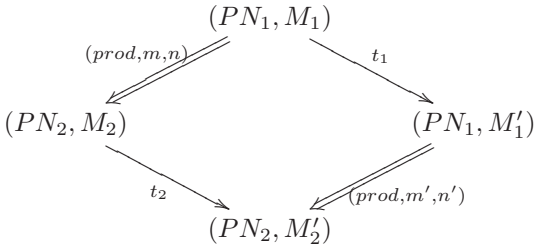


1. $((prod, m, n), t_2)$ is sequentially independent iff $((prod^{-1}, n, m), t_2)$ is parallel independent.
2. $(t_1, (prod, m', n'))$ is sequentially independent iff $((prod, m', n'), t_1^{-1})$ is parallel independent.
3. $(t_2, (prod, m', n'))$ is coparallel independent iff $((prod^{-1}, n', m'), t_2^{-1})$ is parallel independent.

Now we are able to extend Theorem 1 on parallel independence showing that resulting steps in the first diagram of Theorem 1 are sequentially and coparallel independent.

Theorem 2 (Parallel and Sequential Independence). *In Theorem 1, where we start with parallel independence of the upper steps in the following diagram with*

match m and comatch n, we have in addition the following sequential and coparallel independence in the following diagram:



1. *The left pair of steps, short $((prod, m, n), t_2)$, is sequentially independent.*
2. *The right pair of steps, short $(t_1, (prod, m', n'))$, is sequentially independent.*
3. *The lower pair of steps, short $(t_2, (prod, m', n'))$, is coparallel independent.*

Proof. We use the proof of Theorem 1.

1. (a) t_2 is not created because it corresponds to $t_1 \in T_1$ which is not deleted.
   (b) We have $M_R(p) \leq M_2'(n(p))$ for all $p \in P_R$ by construction of the pushout $(2')$ with $M_2'' = M_2'$.
2. (a) t_1 is not deleted by the assumption of parallel independence.
   (b) $M_L(p) \leq M_1(m(p))$ for all $p \in P_L$ by pushout (1).
3. (a) t_2 is not created as shown in the proof of 1. (a).
   (b) $M_R(p) \leq M_2(n(p))$ for all $p \in P_R$ by pushout (2).

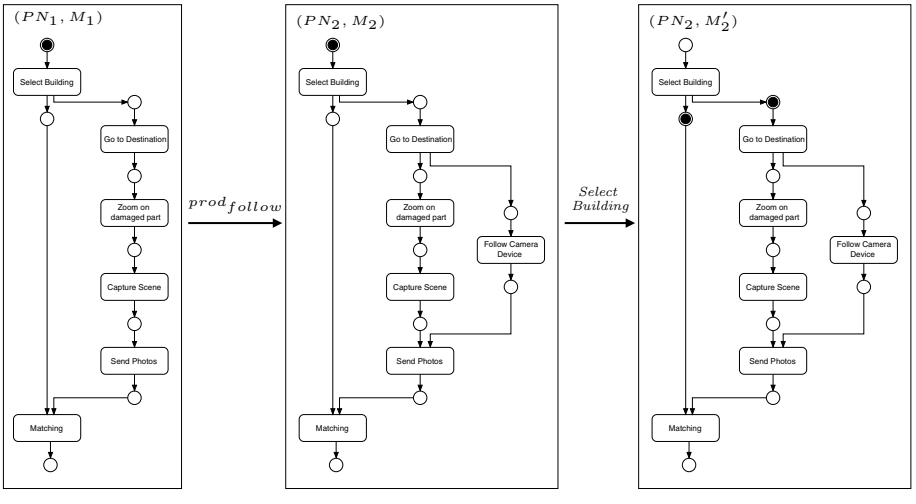
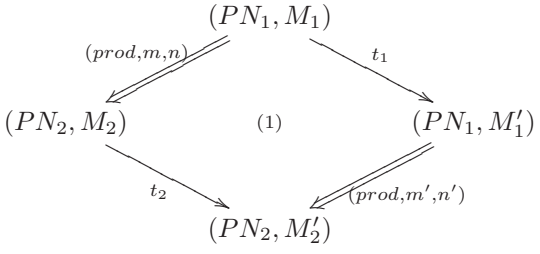


Fig. 6. Pair of steps $((prod_{follow}, m, n)$, *Select Building*$)$

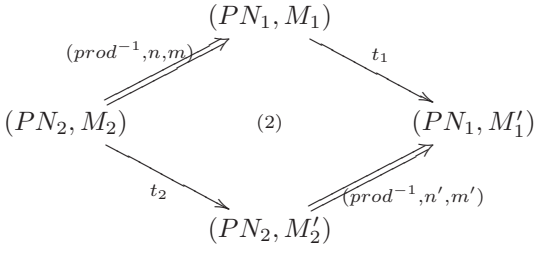
In Theorem 2 we have shown that parallel independence implies sequential and coparallel independence. Now we show vice versa that sequential (coparallel) independence implies parallel and coparallel (parallel and sequential) independence.

Theorem 3 (Sequential and (Co-)Parallel Independence)

1. *Given the left sequentially independent steps in diagram (1) then also the right steps exist, s.t. the upper (right, lower) pair is parallel (sequentially, coparallel) independent.*
2. *Given the right sequentially independent steps in diagram (1) then also the left steps exist, s.t. the upper (left, lower) pair is parallel (sequentially, coparallel) independent.*
3. *Given the lower coparallel independent steps in diagram (1) then also the upper steps exist, s.t. the upper (left,right) pair is parallel (sequentially, sequentially) independent.*



Proof 1. Using Remark 3, left sequential independence in (1) corresponds to parallel independence in (2). Applying Theorem 1 and Theorem 2 to the left pair in (2) we obtain the right pair such that the upper and lower pairs are sequentially and the right pair coparallel independent. This implies by Remark 3 that the upper (right, lower) pairs in (1) are parallel (sequentially, coparallel) independent.



The proofs of items *2.* and *3.* are analogous to the proof of *1.*

5 General Framework of Net Transformations

In [23], we have introduced the paradigm "nets and rules as tokens" using a high-level model with suitable data type part. This model called algebraic higher-order (AHO) system (instead of high-level net and replacement system as in [23])

exploits some form of control not only on rule application but also on token firing. In general an AHO-system is defined by an algebraic high-level net with system places and rule places as for example shown in Fig. 7, where the marking is given by a suitable P/T-system resp. rule on these places. For a detailed description of the data type part, i.e. the AHO-SYSTEM-signature and corresponding algebra A, we refer to [23].

In the following we review the behavior of AHO-systems according to [23]. With the symbol $Var(t)$ we indicate the set of variables of a transition t, i.e., the set of all variables occurring in pre- and post domain and in the firing-condition of t. The marking M determines the distribution of P/T-systems and rules in an AHO-system, which are elements of a given higher-order algebra A. Intuitively, P/T-systems and rules can be moved along AHO-system arcs and can be modified during the firing of transitions. The follower marking is computed by the evaluation of net inscriptions in a variable assignment $v : Var(t) \to A$. The transition t is enabled in a marking M, if and only if (t, v) is consistent, that is if the evaluation of the firing condition is fulfilled. Then the follower marking after firing of transition t is defined by removing tokens corresponding to the net inscription in the pre domain of t and adding tokens corresponding to the net inscription in the post domain of t.

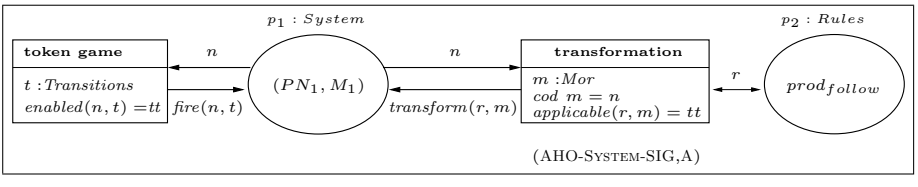


Fig. 7. Algebraic higher-order system

The transitions in the AHO-system in Fig. 7 realize on the one hand firing steps and on the other hand transformation steps as indicated by the net inscriptions $fire(n,t)$ and $transform(r,m)$, respectively. The initial marking is the reconfigurable P/T-system given in Section 2, i.e. the P/T-system (PN_1, M_1) given in Fig. 1 is on the place p_1, while the marking of the place p_2 is given by the rule $prod_{follow}$ given in Fig. 2. To compute the follower marking of the P/T-system we use the transition *token game* of the AHO-system. First the variable n is assigned to the P/T-system (PN_1, M_1) and the variable t to the transition *Select Building*. Because this transition is enabled in the P/T-system, the firing condition is fulfilled. Finally, due to the evaluation of the term $fire(n,t)$ we obtain the new P/T-system (PN_1, M_1') (see Fig. 1).

For changing the structure of P/T-systems the transition *transformation* is provided in Fig. 7. Again, we have to give an assignment v for the variables of the transition, i.e. variables n, m and r, where $v(n) = (PN_1, M_1)$, $v(m)$ is a suitable match morphism and $v(r) = prod_{follow}$. The firing condition $cod\ m = n$ ensures that the codomain of the match morphism is equal to (PN_1, M_1),

while the second condition $applicable(r, m)$ checks the gluing condition, i.e. if the rule $prod_{follow}$ is applicable with match m. Afterwards, the transformation step depicted in Fig. 2 is computed by the evaluation of the net inscription $transform(r, m)$ and the effect of firing the transition *transformation* is the removal of the P/T-system (PN_1, M_1) from place p_1 in Fig. 7 and adding the P/T-system (PN_2, M_2) to it. The pair (or sequence) of firing and transformation steps discussed in the last sections is reflected by firing of the transitions one after the other in our AHO-system. Thus, the results presented in this paper are most important for the analysis of AHO-systems.

6 Conclusion

This paper continues our work on "nets and rules as tokens" [23] by transferring the results of local Church-Rosser, which are well known for term rewriting and graph transformations, to the consecutive evolution of a P/T-system by token firing and rule applications. We have presented conditions for (co-)parallel and sequential independence and we have shown that provided that these conditions are satisfied, firing and transformation steps can be performed in any order, yielding the same result. Moreover, we have correlated these conditions, i.e. that parallel independence implies sequential independence and vice versa, sequential (coparallel) independence implies parallel and coparallel (parallel and sequential) independence. The advantage of the presented conditions is that they can be checked syntactically and locally instead of semantically and globally. Thus, they are also applicable in the case of complex reconfigurable P/T-systems.

Transformations of nets can be considered in various ways. Transformations of Petri nets to another Petri net class (e.g. in [7, 10, 35]), to another modeling technique or vice versa (e.g in [2, 5, 15, 26, 33, 14]) are well examined and have yielded many important results. Transformation of one net into another without changing the net class is often used for purposes of forming a hierarchy, in terms of reductions or abstraction (e.g. in [22, 16, 20, 12, 8]) or transformations are used to detect specific properties of nets (e.g. in [3, 4, 6, 29]). Net transformations that aim directly at changing the net in arbitrary ways as known from graph transformations were developed as a special case of high-level replacement systems e.g. in [17]. The general approach can be restricted to transformations that preserve specific properties as safety or liveness (see [30, 32]). Closely related are those approaches that propose changing nets in specific ways in order to preserve specific semantic properties, as equivalent (I/O-) behavior (e.g in [1, 11]), invariants (e.g. in [13]) or liveness (e.g. in [19, 37]). Related are also those approaches that follow the "nets as tokens"-paradigm, based on elementary object nets introduced in [36]. Mobile object net systems [24, 21] are an algebraic formalization of the elementary object nets that are closely related to our approach. In both cases the data types, respectively the colors represent the nets that are the token nets. Our approach goes beyond those approaches as we additionally have rules as tokens, and transformations of nets as operations. In [24] concurrency aspects between token nets have been investigated, but naturally not

concerning net transformations. In [27] rewriting of Petri nets in terms of graph grammars are used for the reconfiguration of nets as well, but this approach lacks the "nets as tokens"-paradigm.

In this paper we present main results of a line of research[2] concerning formal modeling and analysis of workflows in mobile ad-hoc networks. So, there is a large amount of most interesting and relevant open questions directly related to the work presented here. While a firing step and a transformation step that are parallel independent can be applied in any order, an aspect of future work is under which conditions they can be applied in parallel leading to the notions of parallel steps. Vice versa a parallel step should be splitted into the corresponding firing and transformation steps. This problem is closely related to the Parallelism Theorem for high-level replacement systems [17] which is the basis of a shift construction for transformation sequences. Moreover, it is most interesting to transfer further results which are already valid for high-level replacement systems, e.g. confluence, termination and critical pairs [17]. We plan to develop a tool for our approach using the graph transformation engine AGG[3] as a tool for the analysis of transformation properties like independence and termination, meanwhile the token net properties could be analyzed using the Petri Net Kernel [25], a tool infrastructure for Petri nets different net classes.

References

1. Balbo, G., Bruell, S., Sereno, M.: Product Form Solution for Generalized Stochastic Petri Nets. IEEE Transactions on Software Engineering 28(10), 915–932 (2002)
2. Belli, F., Dreyer, J.: Systems Modelling and Simulation by Means of Predicate/Transition Nets and Logic Programming. In: Proc. Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE), pp. 465–474 (1994)
3. Berthelot, G.: Checking properties of nets using transformation. In: Proc. Applications and Theory in Petri Nets. LNCS, vol. 222, pp. 19–40. Springer, Heidelberg (1985)
4. Berthelot, G.: Transformations and Decompositions of Nets. In: Petri Nets: Central Models and Their Properties, Part I, Advances in Petri Nets. LNCS, vol. 254, pp. 359–376. Springer, Heidelberg (1987)
5. Bessey, T., Becker, M.: Comparison of the modeling power of fluid stochastic Petri nets (FSPN) and hybrid Petri nets (HPN). In: Proc.Systems, Man and Cybernetics (SMC), vol. 2, pp. 354–358. IEEE Computer Society Press, Los Alamitos (2002)
6. Best, E., Thielke, T.: Orthogonal Transformations for Coloured Petri Nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 447–466. Springer, Heidelberg (1997)
7. Billington, J.: Extensions to Coloured Petri Nets. In: Proc. Petri Nets and Performance Models (PNPM), pp. 61–70. IEEE Computer Society Press, Los Alamitos (1989)

[2] The research project *Formal Modeling and Analysis of Flexible Processes in Mobile Ad-hoc Networks* (forMA$_1$NET) of the German Research Council.

[3] tfs.cs.tu-berlin.de/agg

8. Bonhomme, P., Aygalinc, P., Berthelot, G., Calvez, S.: Hierarchical control of time Petri nets by means of transformations. In: Proc. Systems, Man and Cybernetics (SMC), vol. 4, p. 6. IEEE Computer Society Press, Los Alamitos (2002)
9. Bottoni, P., De Rosa, F., Hoffmann, K., Mecella, M.: Applying Algebraic Approaches for Modeling Workflows and their Transformations in Mobile Networks. Journal of Mobile Information Systems 2(1), 51–76 (2006)
10. Campos, J., Sánchez, B., Silva, M.: Throughput Lower Bounds for Markovian Petri Nets: Transformation Techniques. In: Proc. Petri Nets and Performance Models (PNPM), pp. 322–331. IEEE Computer Society Press, Los Alamitos (1991)
11. Carmona, J., Cortadella, J.: Input/Output Compatibility of Reactive Systems. In: Aagaard, M.D., O'Leary, J.W. (eds.) FMCAD 2002. LNCS, vol. 2517, pp. 360–377. Springer, Heidelberg (2002)
12. Chehaibar, G.: Replacement of Open Interface Subnets and Stable State Transformation Equivalence. In: Proc. Applications and Theory of Petri Nets (ATPN). LNCS, vol. 674, pp. 1–25. Springer, Heidelberg (1991)
13. Cheung, T., Lu, Y.: Five Classes of Invariant-Preserving Transformations on Colored Petri Nets. In: Donatelli, S., Kleijn, J.H.C.M. (eds.) ICATPN 1999. LNCS, vol. 1639, pp. 384–403. Springer, Heidelberg (1999)
14. Cortés, L., Eles, P., Peng, Z.: Modeling and formal verification of embedded systems based on a Petri net representation. Journal of Systems Architecture 49(12-15), 571–598 (2003)
15. de Lara, J., Vangheluwe, H.: Computer Aided Multi-Paradigm Modelling to Process Petri-Nets and Statecharts. In: Corradini, A., Ehrig, H., Kreowski, H.-J., Rozenberg, G. (eds.) ICGT 2002. LNCS, vol. 2505, pp. 239–253. Springer, Heidelberg (2002)
16. Desel, J.: On Abstraction of Nets. In: Proc. Applications and Theory of Petri Nets (ATPN). LNCS, vol. 524, pp. 78–92. Springer, Heidelberg (1990)
17. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. In: EATCS Monographs in Theoretical Computer Science, Springer, Heidelberg (2006)
18. Ehrig, H., Hoffmann, K., Prange, U., Padberg, J.: Formal Foundation for the Reconfiguaration of Nets. Technical report, TU Berlin, Fak. IV (2007)
19. Esparza, J.: Model Checking Using Net Unfoldings. Science of Computer Programming 23(2-3), 151–195 (1994)
20. Esparza, J., Silva, M.: On the analysis and synthesis of free choice systems. In: Proc. Applications and Theory of Petri Nets (ATPN). LNCS, vol. 483, pp. 243–286. Springer, Heidelberg (1989)
21. Farwer, B., Köhler, M.: Mobile Object-Net Systems and their Processes. Fundamenta Informaticae 60(1–4), 113–129 (2004)
22. Haddad, S.: A Reduction Theory for Coloured Nets. In *Proc. Applications and Theory in Petri Nets (ATPN).* In: Proc. Applications and Theory in Petri Nets (ATPN). LNCS, vol. 424, pp. 209–235. Springer, Heidelberg (1988)
23. 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)
24. Köhler, M., Rölke, H.: Concurrency for mobile object net systems. Fundamenta Informaticae 54(2-3), 221–235 (2003)
25. Kindler, E., Weber, M.: The Petri Net Kernel - An Infrastructure for Building Petri Net Tools. Software Tools for Technology Transfer 3(4), 486–497 (2001)

26. Kluge, O.: Modelling a Railway Crossing with Message Sequence Charts and Petri Nets. In: Ehrig, H., Reisig, W., Rozenberg, G., Weber, H. (eds.) Petri Net Technology for Communication-Based Systems. LNCS, vol. 2472, pp. 197–218. Springer, Heidelberg (2003)
27. Llorens, M., Oliver, J.: Structural and Dynamic Changes in Concurrent Systems: Reconfigurable Petri Nets. IEEE Transactions on Computers 53(9), 1147–1158 (2004)
28. Meseguer, J., Montanari, U.: Petri Nets Are Monoids. Information and Computation 88(2), 105–155 (1990)
29. Murata, T.: Petri nets: Properties, analysis and applications. In: Proc. IEEE, vol. 77, pp. 541 – 580. IEEE (1989)
30. Padberg, J., Gajewsky, M., Ermel, C.: Rule-based refinement of high-level nets preserving safety properties. Science of Computer Programming 40(1), 97–118 (2001)
31. 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)
32. Padberg, J., Urbášek, M.: Rule-Based Refinement of Petri Nets: A Survey. In: Ehrig, H., Reisig, W., Rozenberg, G., Weber, H. (eds.) Petri Net Technology for Communication-Based Systems. LNCS, vol. 2472, pp. 161–196. Springer, Heidelberg (2003)
33. Parisi-Presicce, F.: A Formal Framework for Petri Net Class Transformations. In: Ehrig, H., Reisig, W., Rozenberg, G., Weber, H. (eds.) Petri Net Technology for Communication-Based Systems. LNCS, vol. 2472, pp. 409–430. Springer, Heidelberg (2003)
34. Rozenberg, G.: Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations. World Scientific, Singapore (1997)
35. Urbášek, M.: Categorical Net Transformations for Petri Net Technology. PhD thesis, Technische Universität Berlin (2003)
36. Valk, R.: Petri Nets as Token Objects: An Introduction to Elementary Object Nets. In: Desel, J., Silva, M. (eds.) ICATPN 1998. LNCS, vol. 1420, pp. 1–25. Springer, Heidelberg (1998)
37. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems and Computers 8(1), 21–66 (1998)