Electronic Communications of the EASST
Volume 10 (2008)

Proceedings of the
Seventh International Workshop on
Graph Transformation and Visual Modeling Techniques
(GT-VMT 2008)

Negative Application Conditions for Reconfigurable Place/Transition Systems

A. Rein, U. Prange, L. Lambers, K. Hoffmann, J. Padberg

14 pages

Guest Editors: Claudia Ermel, Reiko Heckel, Juan de Lara
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Negative Application Conditions for Reconfigurable Place/Transition Systems

A. Rein, U. Prange, L. Lambers, K. Hoffmann, J. Padberg

Technische Universität Berlin
Institut für Softwaretechnik und Theoretische Informatik

Abstract: This paper introduces negative application conditions for reconfigurable place/transition nets. These are Petri nets together with a set of rules that allow changing the net and its marking dynamically. Negative application conditions are a control structure that prohibits the application of a rule if certain structures are already existent. We motivate the use of negative application conditions in a short example. Subsequently the underlying theory is sketched and the results – concerning parallelism, concurrency and confluence – are presented. Then we resume the example and explicitly discuss the main results and their usefulness within the example.

Keywords: Petri net, net transformation, control structure, negative application condition

1 Introduction

As the adaptation of a system to a changing environment gets more and more important, Petri nets that can be transformed during runtime have become a significant topic in recent years. Application areas cover, among others, computer supported cooperative work, multi-agent systems, dynamic process mining and mobile networks. Moreover, this approach increases the expressiveness of Petri nets and allows a formal description of dynamic changes. In [HEM05], this concept of reconfigurable place/transition (P/T) systems has been introduced where the main idea is the stepwise development of P/T systems by rules where the left-hand side is replaced by the right-hand side preserving a context. We use rules and transformations for place/transition systems in the sense of the double pushout approach for graph transformation (see [EP04]). More precisely, adhesive high-level replacement (HLR) systems – a suitable categorical framework for double pushout transformations [EEPT06] – have been instantiated to P/T systems.

For the suitable application of such rules specific control structures are needed, especially the possibility to forbid certain rule applications. These are known from graph transformation systems as negative application conditions (NACs). 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 replacement system.

By proving that P/T systems are weak adhesive HLR categories with negative application conditions we can transfer well-known and important results to this case: Local Church-Rosser Theorem, Completeness Theorem for Critical Pairs, Concurrency Theorem, Embedding and Extension Theorem and Local Confluence Theorem or Critical Pair Lemma.

This paper is organized as follows: first we introduce our example and discuss the need of additional control structures for the application of rules in Section 2. Then we review the formal notions for reconfigurable P/T systems in Section 3. Based on these notions we define negative application conditions and present the main results concerning parallelism, concurrency and confluence in Section 4. We discuss some of the general results with respect to the example in Section 2. Concluding remarks concern future and related work.

2 Example: Airport

In this section, we present an example for a reconfigurable P/T system with NACs. We model an airport control system (AirCS) that organizes the starting and landing runways of an airport. The P/T system has to ensure that certain safety properties of an airport are fulfilled, for example that some areas of the airport like the actual runways are secure, i.e. exclusively used by one airplane at the time. The AirCS is able to adapt to various changes of the airport. Changes at runtime may concern the opening or closing of a runway or its kind of use, i.e. for starting or landing. As a basic condition we require that there has to be at least one landing runway to ensure the landing of arriving airplanes, especially in emergency situations.

In addition, the use of the starting runways depends on the weather. Under fair weather conditions, no limitations occur. But it is possible for the system to receive a storm warning from a weather information channel. In this case, no more starting runways shall be opened and when the storm arrives it shall be forbidden for airplanes to depart.

In the top of Fig. 1, the standard AirCS with one starting and one landing runway is depicted. Each runway consists of two places of type Runway and Tower, with exactly one token on either the one or the other place. A token on Runway represents an airplane on this runway, while a token on Tower means that this runway is currently not in use. In addition, a landing runway consists of transitions landing and arrived, and a starting runway of transitions depart and takeoff, respectively. By firing the transition approach an airplane appears in the airspace of the AirCS. The transition landing may fire if the runway is currently not used leading to a token representing an airplane on the runway. In the lower part of the P/T system, the gates area is modeled. The place Gates is a counter for the available gates. If a gate is available, the airplane may proceed to a gate by firing the transition arrived. If the deboarding process is completed, the firing of the transition continueFlight initiates the boarding process, and using the transitions depart and takeoff an airplane may depart over an available starting runway. In contrast to transition approach, the firing of the transition quitting models that the airplane leaves the airspace of the AirCS.

In the following, we describe the rules for changes of the airport. The rules openStartingRunway and openLandingRunway in the top of Fig. 2 are used to open a new runway. The places and transitions of the runway are inserted and connected with the already existing part of the airport. For the rule openStartingRunway, an additional NAC is needed to prevent the opening of a starting runway in case of a storm warning. To apply a rule to our P/T system, we first have to find a match of the left-hand side L to the P/T system. In case of the rule openStartingRunway, this match is unique and consists of the four places in L. If we have a NAC for the rule, we have to check if this NAC is valid, which means that we are not allowed to find a morphism from the

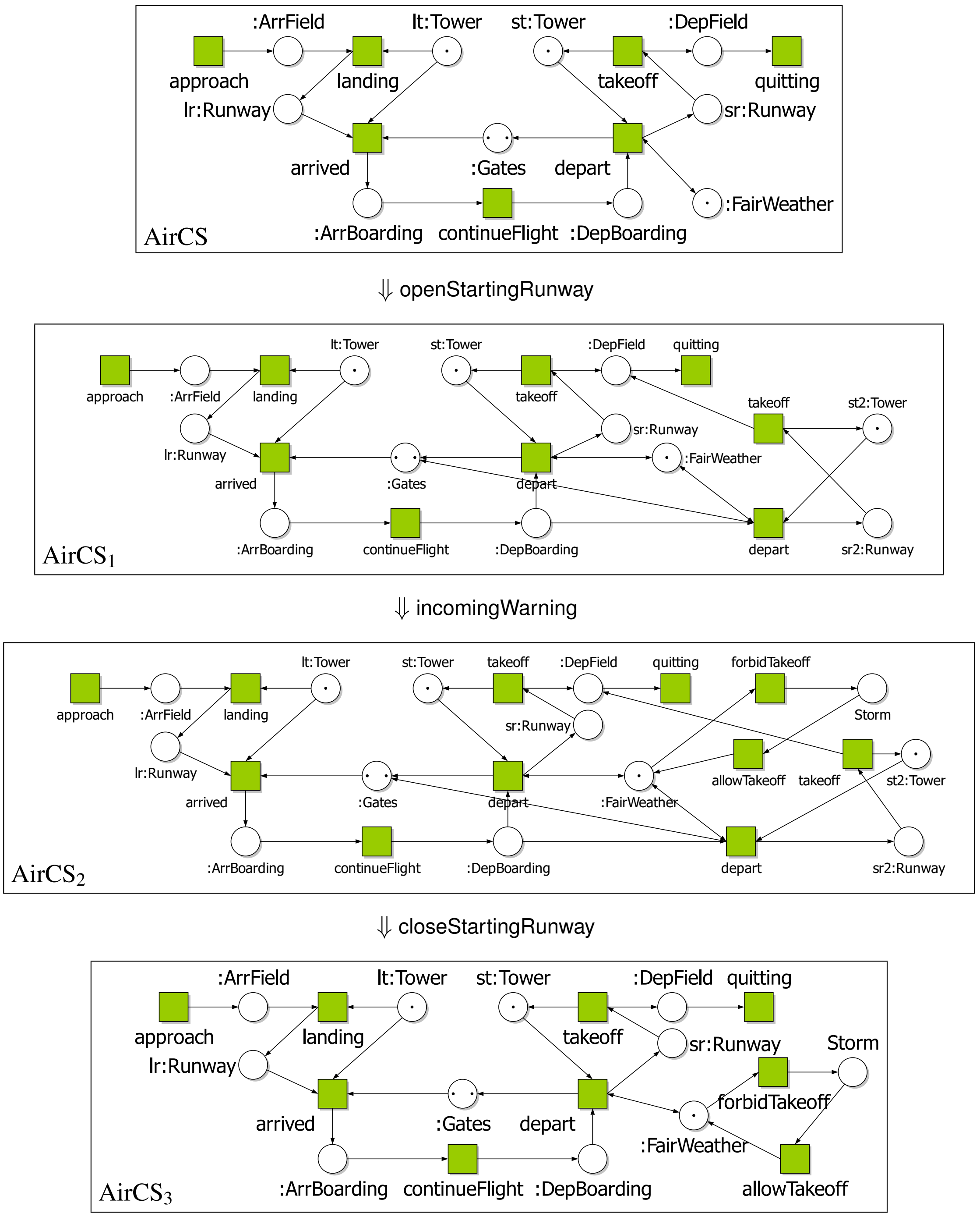

Figure 1: The standard AirCS and a transformation sequence

NAC to our P/T system via the match. For the match of openStartingRunway, the NAC is fulfilled as long as no place of type Storm exists. In addition, a gluing condition has to be fulfilled to make sure the rule is applicable (see Def. 2 in Section 3). Then, we first delete those elements that are no longer needed and insert those elements that shall be created. They are glued to the already existing elements, for example when applying openStartingRunway the new transitions are connected to the places in the match. The application of the rule openStartingRunway to the AirCS from Fig. 1 is the P/T system AirCS$_1$ in the upper middle of Fig. 1.

The rules closeStartingRunway and closeLandingRunway in the upper middle of Fig. 2 are used to close a runway. closeStartingRunway is the inverse rule of openStartingRunway without a NAC, since closing a starting runway is always allowed. For closeLandingRunway, we have to make sure that we do not delete the last landing runway, thus another landing runway has to be present in the match.

In the lower middle of Fig. 2, the rules changeLandingToStartingRunway and changeStartingToLandingRunway for changing the kind of a runway are shown. For both rules, we have a NAC that forbids that the runway is currently used by an airplane, represented by a token on the runway which is not present in the left-hand side. This ensures that the type of the runway is not changed during its use causing strange behaviour for incoming or outgoing flights. In addition, there is a second NAC that forbids the application of changeLandingToStartingRunway in case of a storm. In the NACs, the place types are omitted if they are obvious from the left-hand side.

The arriving of a storm warning is also modeled by a rule as shown in the bottom of Fig. 2. A new place of type Storm is created and two transitions between this place and the place FairWeather. As soon as the storm warning has arrived, no starting runways can be opened. But it is still possible to take off for waiting airplanes using the already existing starting runways. The airport system itself can decide when the weather situation is that bad that no airplan shall depart. Then the transition forbidTakeoff is fired and since there is no longer a token on the place FairWeather, no more takeoffs are possible. As soon as the weather gets better, by firing the transition allowTakeoff airplanes may depart. Then the rule clearWarning deletes the storm warning and the regular airport operations may continue.

Altogether, in Fig. 1 a transformation sequence is depicted which first opens a starting runway, then receives a storm warning and afterwards, the starting runway is closed again.

3 Reconfigurable Place/Transition Nets

In this section, we formalize reconfigurable P/T systems based on our results in [HEM05]. As net formalism we use the algebraic notation of “Petri nets are Monoids” in [MM90], but extend this notation by a label function for places. So, a P/T net is given by $PN = (P, T, pre, post, label)$ with pre- and post domain functions $pre, post : T \to P^{\oplus}$ and a label function $label : P \to L$, where L is a fixed alphabet for places and $P^{\oplus}$ is the free commutative monoid over the set P of places, and a P/T system is given by (PN, M) with marking $M \in P^{\oplus}$.

In order to define rules and transformations of P/T systems we introduce P/T morphisms which preserve firing steps by Condition (1) and labels by Condition (2) below. Additionally, they require that the initial marking at corresponding places is increasing (Condition (3)). For strict morphisms, in addition injectivity and the preservation of markings is required (Condition (4)).

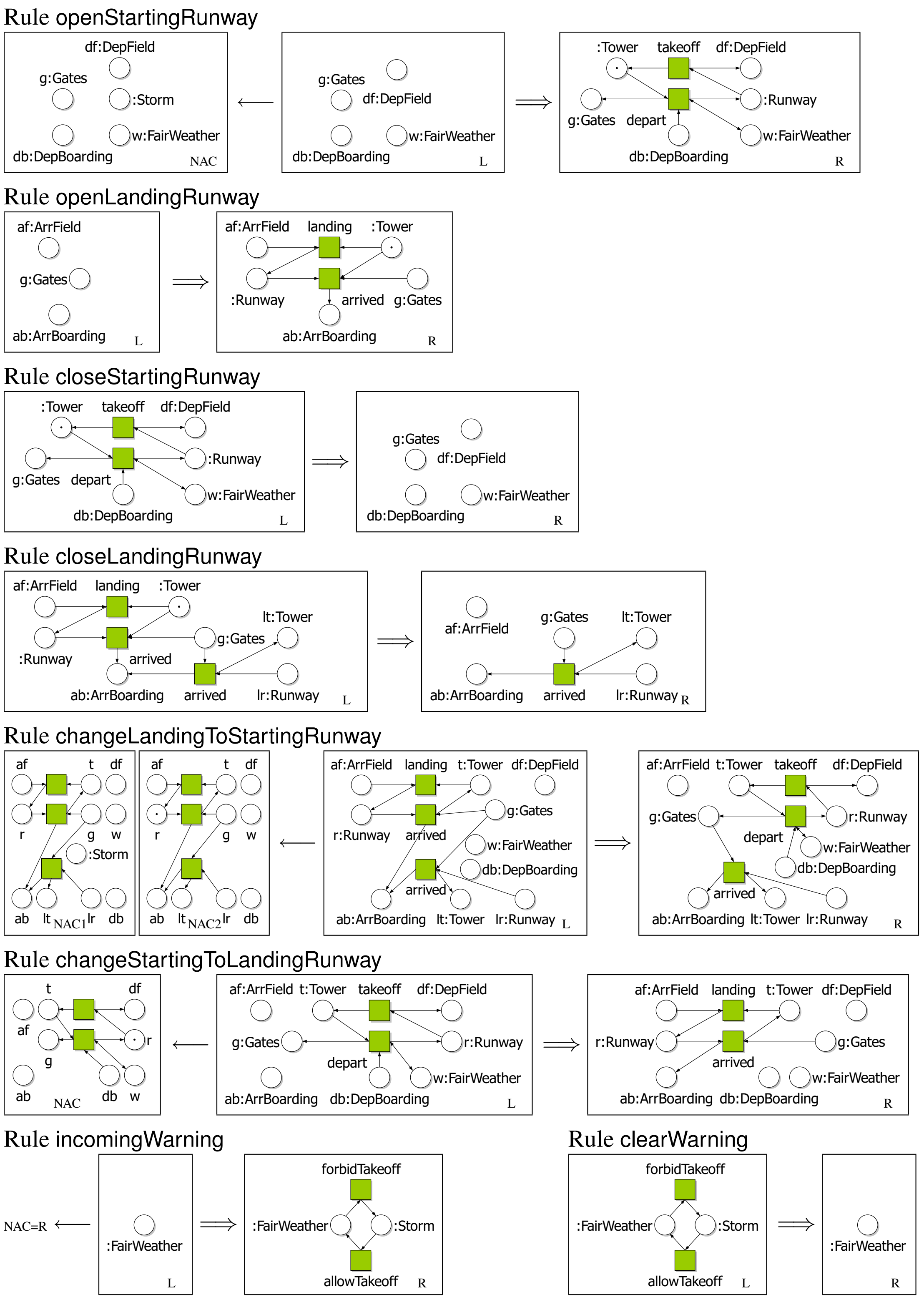

Figure 2: The rules for changing the AirCS

Definition 1 (P/T Morphism) Given P/T systems $PS_i = (P_i, T_i, pre_i, post_i, label_i, M_i)$ for $i = 1,2$, a P/T morphism $f : PS_1 \to PS_2$ 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 $f_P^{\oplus} \circ post_1 = post_2 \circ f_T$,
(2) $label_1 = label_2 \circ f_P$ and
(3) $M_1(p) \leq M_2(f_P(p))$ for all $p \in P_1$.

Moreover, the P/T morphism f is called strict if **(4)** f_P and f_T are injective and $M_1(p) = M_2(f_P(p))$ for all $p \in P_1$.
The category defined by P/T systems and P/T morphisms is denoted by **PTS** where the composition of P/T morphisms is defined component-wise for places and transitions. The class of all strict P/T morphisms is denoted by $\mathscr{M}$.

Next we define the 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 which is needed for the first step in a transformation.

Definition 2 (Gluing Condition) Given P/T systems $PS_i = (P_i, T_i, pre_i, post_i, label_i, M_i)$ for $i \in \{L, K, 1\}$, and let $PS_L \xrightarrow{m} PS_1$ be a P/T morphism and $PS_K \xrightarrow{l} PS_L$ a strict morphism, then the gluing points GP, the dangling points DP and the identification points IP of PS_L are defined by

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

The P/T morphisms m and l with l strict 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. $\forall p \in P_L \setminus l(P_K) : M_L(p) = M_1(m(p))$.

Next we present rule-based transformations of P/T systems following the double-pushout (DPO) approach of graph transformations in the sense of [Roz97, EEPT06].

Definition 3 (P/T System Rule) Given P/T systems $PS_i = (P_i, T_i, pre_i, post_i, label_i, M_i)$ for $i \in \{L, K, R, 1\}$, then a rule $rule = (PS_L \xleftarrow{l} PS_K \xrightarrow{r} PS_R)$ consists of P/T systems PS_L, PS_K, and PS_R, called the left-hand side, interface, and right-hand side of $rule$, respectively, and two strict P/T morphisms $PS_K \xrightarrow{l} PS_L$ and $PS_K \xrightarrow{r} PS_R$.
The rule $rule$ is applicable at the match $PS_L \xrightarrow{m} PS_1$ if the gluing condition is satisfied for l and m. In this case, we obtain a P/T system PS_0 leading to a transformation step $PS_1 \overset{rule,m}{\Longrightarrow} PS_2$ consisting of the following pushout diagrams (1) and (2). The P/T morphism $n : PS_R \to PS_2$ is called comatch of the transformation step.

$$\begin{tikzcd}
PS_L \arrow[d, "m"'] \arrow[dr, phantom, "(\mathbf{1})"] & PS_K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "c"] \arrow[dr, phantom, "(\mathbf{2})"] & PS_R \arrow[d, "n"] \\
PS_1 & PS_0 \arrow[l, "l^*"] \arrow[r, "r^*"'] & PS_2
\end{tikzcd}$$

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

Definition 4 (Reconfigurable P/T Systems) Given a P/T system PS and a set of rules $RULES$, a reconfigurable P/T system is defined by $(PS, RULES)$.

In the example in Section 2 the reconfigurable P/T system consists of the P/T system in the top of Fig. 1 and the set of rules depicted in Fig. 2. Note, that the application of some of these rules is restricted by NACs and we will present the notion of reconfigurable P/T systems with NACs in Section 4.

4 Negative Application Conditions

In this section, we first state the main technical result that P/T systems are a weak adhesive HLR category with NACs. As a consequence we can define NACs for P/T system rules and transformations. Afterwards we summarize the main results available for reconfigurable P/T systems with NACs.

In addition to the class $\mathcal{M}$ in Section 3 we need two other classes of morphisms. The class $\mathcal{Q}$ denotes those morphisms that connect the NAC to the source net, which is the class of injective P/T system morphisms. Note that morphisms in this class do not need to be marking strict. The class $\mathcal{E}$ is a class of minimal jointly surjective morphism pairs, where minimal means that the markings in the codomain are as small as possible, i.e. for $e_1 : PS_1 \to PS_3$, $e_2 : PS_2 \to PS_3$ with $(e_1, e_2) \in \mathcal{E}$ we have that e_1, e_2 are jointly surjective and $M_3(p) = \max(\{M_1(p') \mid p' \in e_1^{-1}(p)\} \cup \{M_2(p') \| \; p' \in e_2^{-1}(p)\})$. This class is mainly used for constructions and proofs.

Definition 5 (Morphism classes in **PTS**) Given the category **PTS** of P/T systems and P/T morphisms, then the following morphism classes are defined:

$\mathcal{M}$	:	strict **PTS** morphisms (injective and marking strict **PTS** morphisms)
$\mathcal{Q}$	:	injective **PTS** morphisms (monomorphisms in the category **PTS**)
$\mathcal{E}$	:	minimal jointly surjective **PTS** morphisms

Theorem 1 (**PTS** is weak adhesive HLR category with NACs) *Given the weak adhesive HLR category* **PTS** *and the morphism classes $\mathcal{M}$, $\mathcal{Q}$ and $\mathcal{E}$ as defined above then we have*

1. *unique $\mathcal{E}$-$\mathcal{Q}$ pair factorization,*
2. *unique epi-$\mathcal{M}$ factorization,*
3. *$\mathcal{M}$-$\mathcal{Q}$ pushout-pullback decomposition property,*

4. *initial pushouts over $\mathcal{Q}$-morphisms,*

5. *$\mathcal{Q}$ is closed under pushouts and pullbacks along $\mathcal{M}$-morphisms,*

6. *induced pullback-pushout property for $\mathcal{M}$ and $\mathcal{Q}$ and*

7. *$\mathcal{Q}$ is closed under composition and decomposition.*

Altogether, $(\mathbf{PTS}, \mathcal{M}, \mathcal{E}, \mathcal{Q})$ is a weak adhesive HLR category with NACs.

Proof. In [EEH$^+$07], it has been shown that $(\mathbf{PTS}, \mathcal{M})$ is a weak adhesive HLR category. According to [LEOP08], for a weak adhesive HLR category with NACs Items 1–7 have to be proven additionally. Note, that in [LEOP08] an additional morphism class $\mathcal{M}'$ is used and some more properties have to be checked. In the case of **PTS**, $\mathcal{Q}$ and $\mathcal{M}'$ coincide, which reduces the effort for the proof. Here we only explain the properties and give proof ideas, the detailed proof can be found in [Rei08].

1. unique $\mathcal{E}$-$\mathcal{Q}$ pair factorization: For a morphism pair $f_1 : L_1 \to P$, $f_2 : L_2 \to P$, an $\mathcal{E}$-$\mathcal{Q}$ pair factorization is a pair $e_1 : L_1 \to K$, $e_2 : L_2 \to K$ with $(e_1, e_2) \in \mathcal{E}$ and $m : K \to P \in \mathcal{Q}$ such that $m \circ e_1 = f_1$ and $m \circ e_2 = f_2$. Uniqueness means that two $\mathcal{E}$-$\mathcal{Q}$ pair factorizations of f_1 and f_2 are isomorphic.

   For the construction of this $\mathcal{E}$-$\mathcal{Q}$ pair factorization in **PTS**, we first construct the coproduct L of L_1 and L_2 with coproduct inclusions i_1 and i_2, obtain a morphism $f : L \to P$ and construct an epi-mono factorization $(e : L \to K, m : K \to P)$ of f in P/T nets. Defining $M_K(p) = \max(\{M_L(p') \mid p' \in e^{-1}(p)\})$ and $e_1 = e \circ i_1, e_2 = e \circ i_2$ leads to a unique $\mathcal{E}$-$\mathcal{Q}$ pair factorization. This property is needed mainly for the embedding and extension of transformation pairs.

2. unique epi-$\mathcal{M}$ factorization: For a morphism $f : L \to P$, an epi-$\mathcal{M}$ factorization is an epimorphism $e : L \to K$ and a morphism $m : K \to P \in \mathcal{M}$ such that $m \circ e = f$. Uniqueness means that two epi-$\mathcal{M}$ factorizations of f are isomorphic.

   For the construction, we use the epi-mono factorization of f in P/T nets, and obtain the marking from the marking of P leading to a strict morphism $m \in \mathcal{M}$. Uniqueness follows directly from uniqueness of the epi-mono factorization in P/T nets and strictness of $\mathcal{M}$. This property is needed for the translation of NACs over a morphism.

3. $\mathcal{M}$-$\mathcal{Q}$ pushout-pullback decomposition property: Given the following commutative diagram with $l \in \mathcal{M}$ and $w \in \mathcal{Q}$, where $(1+2)$ is a pushout and (2) is a pullback, then (1) and (2) are both pushouts.

   In P/T nets, this property holds for injective l and w, thus we obtain pushouts (1) and (2) in P/T nets. It remains to show the additional pushout properties in **PTS**, which can be verified. This property is needed for the embedding and extension of transformation pairs and for the equivalence of left and right NACs.

4. initial pushouts over $\mathcal{Q}$-morphisms: An initial pushout over a morphism $f \in \mathcal{Q}$ represents the boundary and context of f. The construction is similar to that in P/T nets, but also

includes all places where f is not marking-strict. This property is needed for the extension of transformations.

5. $\mathscr{Q}$ is closed under pushouts and pullbacks along $\mathscr{M}$-morphisms: In a pushout or pullback square along $\mathscr{M}$, if the other given morphism is in $\mathscr{Q}$ then the opposite morphism is also a $\mathscr{Q}$-morphism.

   This property follows directly from the corresponding properties in P/T nets and is used for the translation of NACs as well as for the embedding and extension of transformations.

6. induced pullback-pushout property for $\mathscr{M}$ and $\mathscr{Q}$: Given the following pushout (PO) and the pullback (PB) then the induced morphism $x : PS_3 \to PS_4$ is a $\mathscr{Q}$-morphism.

   This property can be shown using the fact that f' and g' are jointly surjective, which can be shown for the pushout construction. It is needed for the translation of NACs over a morphism.

7. $\mathscr{Q}$ is closed under composition and decomposition: This is a standard result for monomorphisms from category theory. This property is needed for the translation of NACs and for the completeness of critical pairs.

$$\begin{tikzcd}
A \arrow[r, "k"] \arrow[d, "l"'] \arrow[dr, phantom, "(1)"] & B \arrow[r, "r"] \arrow[d, "s"] \arrow[dr, phantom, "(2)"] & E \arrow[d, "v"] \\
C \arrow[r, "u"] & D \arrow[r, "w"] & F
\end{tikzcd}
\quad
\begin{tikzcd}
PS_0 \arrow[r, "f"] \arrow[d, "g"'] \arrow[dr, phantom, "(PB)"] & PS_1 \arrow[d, "h"] \\
PS_2 \arrow[r, "h'"] & PS_4
\end{tikzcd}
\quad
\begin{tikzcd}
PS_0 \arrow[r, "f"] \arrow[d, "g"'] \arrow[dr, phantom, "(PO)"] & PS_1 \arrow[d, "g'"] \\
PS_2 \arrow[r, "f'"] & PS_3
\end{tikzcd}$$ □

Now, we can state negative application conditions for P/T system transformation in the following sense:

Definition 6 ((Left) Negative Application Condition) A (left) negative application condition of a rule $rule = (PS_L \overset{l}{\leftarrow} PS_K \overset{r}{\rightarrow} PS_R)$ in the weak adhesive HLR category with NACs $(C, \mathscr{M}, \mathscr{E}, \mathscr{Q})$ is of the form $NAC(n)$, where $n : PS_L \to PS_N$ is a P/T morphism.
A morphism $m : PS_L \to PS_1$ satisfies $NAC(n)$, written $m \models NAC(n)$, if there does not exist a morphism $q : PS_N \to PS_1 \in \mathscr{Q}$ with $q \circ n = m$.

Definition 7 (Rule with NACs) A rule in a weak adhesive HLR category with NACs $(C, \mathscr{M}, \mathscr{E}, \mathscr{Q})$ with a set of negative application conditions $NACS$ is called rule with NACs.

Remark 1 *Analogously to left NACs we can define right NACs on the right-hand side of a rule which have to be satisfied by the comatch of the transformation. In this paper, we only consider rules with an empty set of right NACs. This is without loss of generality since each right NAC can be translated into an equivalent left NAC as shown in [EEPT06, LEOP08].*

Definition 8 (Applicability of a Rule with NACs) Given a rule $rule = (PS_L \overset{l}{\leftarrow} PS_K \overset{r}{\rightarrow} PS_R)$ with a set of negative application conditions $NACS$ and a match $m : PS_L \to PS_1$ such that $rule$ without NACs is applicable at m, then the rule $rule$ with NACs is applicable if and only if m satisfies all NACs of the set $NACS$.

For these new rules and their restricted application we obtain the same results as known for net transformations in general. These results have been shown for NACs at the level of weak adhesive HLR categories in [LEOP08, LEO06, Lam07]. Their instantiation to P/T systems requires Theorem 1 above.

Results (For Reconfigurable P/T Systems with NACs)

1. **Local Church-Rosser and Parallelism:** The local Church-Rosser property for transformations with NACs states that for two rules with two matches, the application of the rules at the matches to the same P/T system in any order (sequentially independence) yields the same result if and only if the transformations are parallel independent. For parallel independence of two transformations with NACs it has to be checked in particular if one transformation does not delete anything the other transformation needs, and, in addition, that one transformation does not produce any structure that is forbidden by the other one. For such independent transformations a parallel transformation with NACs can be built obtaining the same result in one transformation step.

   In our airport example it makes e.g. no difference if a starting or a landing runway is opened first. After opening a starting runway it is still possible to open a landing runway, since nothing is deleted and there is no NAC on the rule openLandingRunway. After opening a landing runway a starting one can be opened: Nothing is deleted and the NAC of the rule openStartingRunway that forbids the existence of a place of type Storm remains satisfied. Moreover now both runways can be constructed in parallel. This construction leads to the same result consisting of an airport with one more starting and one more landing runway.

2. **Conflicts and Critical Pairs:** If two transformations are not parallel independent as described in Item 1 then they are in conflict. This means in particular that one of the transformations deletes some structure which is needed by the other one, or it produces a structure which is forbidden by the other one. A critical pair describes such a conflict between two transformations in a minimal context. Critical pairs are proven to be complete [LEO06], i.e. each conflict occurring in the system between two transformations is represented by a critical pair expressing the same conflict in a minimal context. The morphism class $\mathscr{E}$ is required to express this minimal context.

   In our example a transformation adding a warning (i.e. applying the rule incomingWarning) is in conflict with a transformation which opens a starting runway (i.e. applying rule openStartingRunway). This conflict is caused by the NAC of the rule openStartingRunway as it cannot be applied if a place of type Storm is present. This conflict occurs regardless of the number of runways already present in the airport and is expressed in a minimal context by the critical pair shown in Figure 3.

3. **Concurrency:** As explained in Item 1 sequentially independent transformations can be put into one parallel transformation step having the same effect. But if sequential dependencies occur between direct transformations in a transformation sequence the Parallelism Theorem cannot be applied. In this case a so-called concurrent rule with NACs can be constructed establishing the same effect in one transformation step with NACs as the whole

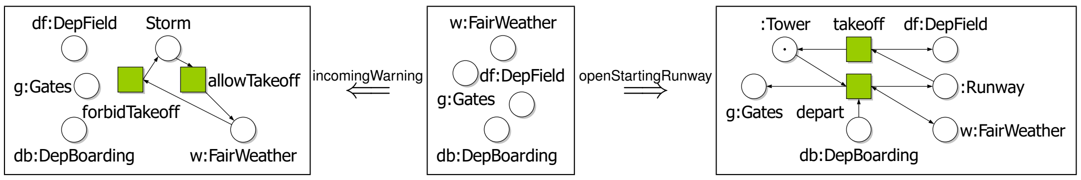

Figure 3: The critical pair for the rules incomingWarning and openStartingRunway

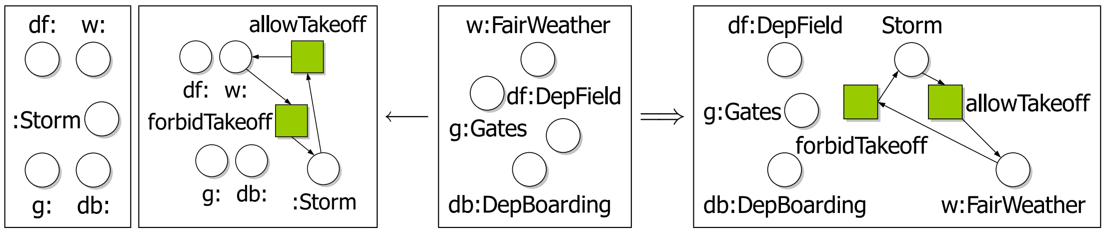

Figure 4: The concurrent rule with NACs

transformation sequence. The Concurrency Theorem states that a concurrent rule with NACs equivalent to a sequence of rules with NACs is applicable with the same result if and only if the rule sequence with NACs is applicable. The construction of the concurrent rule is analogous to the case without NACs. Additionally, all NACs occurring in the rule sequence need to be translated into equivalent NACs for the concurrent rule. The construction of such a concurrent rule with NACs is explained in [LEOP08]. A concurrent rule summarizes in one rule which parts of the net should be present, preserved, deleted and produced when applying the corresponding rule sequence to this net. Moreover we have a summarized set of NACs on the concurrent rule expressing which net parts are forbidden when applying the corresponding rule sequence with NACs to the net.

Consider for our example the transformation sequence in Fig. 1. First, a starting runway is opened followed by an incoming warning after which the starting runway is closed again. This transformation sequence can be summarized to one transformation step via a new concurrent rule with concurrent NACs as depicted in Fig. 4. Note that this concurrent rule now holds two single NACs, one originating from the first rule openStartingRunway and the other one originating from the second rule incomingWarning in the sequence. Note, moreover, that this rule adds no new behavior to our system, but merely adds the possibility of performing these three transformations in one step with the same result.

4. **Embedding and Extension:** Consider a transformation $t : N_0 \overset{*}{\Rightarrow} N_n$ and a morphism $k_0 : N_0 \to N_0'$, then the transformation t can be embedded into the larger context N_0' if and only if the extension morphism $k_0 : N_0 \to N_0'$ satisfies two consistency conditions. First, it has to be boundary consistent. This means intuitively that the extension morphism cannot embed places which are deleted by the transformation t into places connected with new transitions in the bigger P/T system N_0'. Otherwise, dangling edges will occur during the embedding. Moreover, the extension morphism k_0 should satisfy NAC-consistency

[LEO06]. Intuitively, the transformation t can be summarized into one step $t_c : N_0 \Rightarrow N_n$ using the new concurrent rule with concurrent NACs. Now the extension morphism k_0 may not map to a larger P/T system N_0' with added structures which are forbidden by this concurrent NAC. Note that boundary consistency and NAC-consistency are not only sufficient, but also necessary conditions for the construction of extended transformations with NACs. So, whenever it is possible to repeat a transformation t into a bigger context N_0' the extension morphism was boundary and NAC-consistent.

In our example, we can embed the transformation described in the last item into the P/T system AirCS$_1$ from Fig. 1 which is not the initial one, but already contains two starting and one landing runway. On the contrary, this transformation cannot be embedded into the airport AirCS$_2$ from Fig. 1 which already contains a storm warning. This is because the extension morphism k_0 adds a place of type Storm which is forbidden by the concurrent NACs as depicted in Fig. 4.

5. **Confluence:** Critical pairs as described in Item 2 are not only complete, their confluence behavior has an impact on the confluence behavior of the whole system. Intuitively this means that if a conflict can be resolved in a certain way in its minimal context, the same conflict is resolvable as well if it occurs in a larger context. A solution of a conflict in a minimal context (or critical pair) $P_1 \leftarrow K \rightarrow P_2$ is a pair of transformation sequences t_1 and t_2 such that t_1 transforms P_1 into a certain net X and t_2 transforms it into the same net X. Thus the same system state can be reached again if a conflict occurred. The solution of the critical pair needs to be strict. Intuitively speaking, this means that t_1 and t_2 preserve everything which is preserved in common by the critical pair itself. Moreover, whenever it is possible to embed the critical pair into some larger context the extension morphism should be NAC-consistent with respect to the critical pair solution (NAC-confluence [Lam07]). This means in particular that all NACs occurring in the solution of the critical pair are still satisfied by the embedding into the larger context. Under these conditions, this conflict can be resolved in the same way also in a larger context. Otherwise, the solution of the critical pair is no solution for the larger context, and no prediction for confluence can be infered. In particular, if all critical pairs of the reconfiguration system are strictly NAC-confluent then the system is locally confluent.

   Consider in our example the critical pair depicted in Fig. 3. The solution of the critical pair is depicted in Fig. 5 for the right-hand side of the critical pair. Whenever a starting runway has been opened, a warning can come in and the starting runway can be closed again. The result is a P/T system containing no runway, but a storm warning. This is already our solution because this P/T system is identical to the the first P/T system in the critical pair. This solution is moreover strictly confluent because the places g, df, fw and db are preserved by both the critical pair and our solution. Moreover, the solution is NAC-confluent because the satisfaction of the concurrent NACs as depicted in Fig. 4 is implied already by the satisfaction of the first NAC of the critical pair. This means in effect that if the critical pair can be embedded and thus the NACs of the critical pair are satisfied by this embedding then they will also be satisfied when embedding the solution into the same bigger context. Therefore, it is possible to resolve the conflict between an incoming

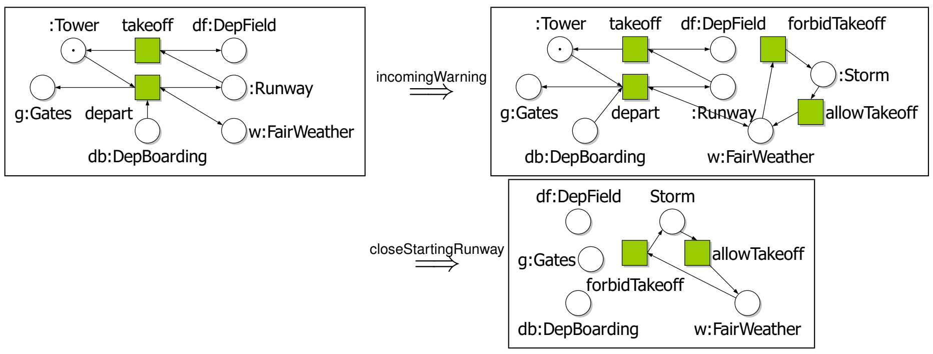

Figure 5: The solution for the critical pair

warning and opening a starting runway by having an incoming warning and closing the runway in each bigger airport as well.

5 Conclusion

We conclude with a short discussion of related and future work:
Related Work Reconfigurable nets have been defined based on net transformations that aim directly at changing the net in arbitrary ways. This approach can be restricted to transformations that preserve specific properties as safety or liveness (see [PU03]). Dynamic nets [BS01] are based on the join calculus and allow the dynamic adaption of the network configuration and are considered to be a special case of zero-safe nets [BMM04]. In a series of papers [LO04, LO06a, LO06b] rewriting of Petri nets in terms of graph grammars is used for the reconfiguration of nets as well. These marked-controlled reconfigurable nets (MCRN) are extended by some control technique that allows changes of the net for specific markings. The enabling of a rule is not only dependent on the net topology, but also dependent on the marking of specific control places. *MCReNet* [LO06a] is the corresponding tool for the modeling and verification of MCRNs.
Future Work One ongoing research task is the extension of this paper's results to algebraic high-level nets, a Petri net variant with additional data types in terms of algebraic specifications. Therefore, the same conditions have to be proved considering additionally the specification and algebra morphisms. Another one are algebraic higher order (AHO) nets that can be used as a controlling mechanism for reconfigurable Petri nets. AHO nets have dynamical tokens like Petri systems as well as transformation rules. This specification technique has been targeted at modeling workflows of mobile ad-hoc networks. Up to now we have not made use of the new feature of NACs in AHO nets. To do so we have to integrate the NACs into the algebra underlying the AHO net.

References

[BMM04] R. Bruni, H. Melgratti, U. Montanari. Extending the Zero-Safe Approach to Coloured, Reconfigurable and Dynamic Nets. In *Lectures on Concurrency and Petri*

Nets. LNCS 3098, pp. 291–327. Springer, 2004.

[BS01] M. Buscemi, V. Sassone. High-Level Petri Nets as Type Theories in the Join Calculus. In *Proceedings of FoSSaCS '01*. Springer, 2001.

[EEH$^+$07] H. Ehrig, C. Ermel, K. Hoffmann, J. Padberg, U. Prange. Concurrency in Reconfigurable Place/Transition Systems: Independence of Net Transformations as well as Net Transformations and Token Firing. Technical report 2007-02, TU Berlin, 2007.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs. Springer, 2006.

[EP04] H. Ehrig, J. Padberg. Graph Grammars and Petri Net Transformations. In *Lectures on Concurrency and Petri Nets*. LNCS 3098, pp. 496–536. Springer, 2004.

[HEM05] K. Hoffmann, H. Ehrig, T. Mossakowski. High-Level Nets with Nets and Rules as Tokens. In *Proceedings of ICATPN'05*. LNCS 3536, pp. 268–288. Springer, 2005.

[Lam07] L. Lambers. Adhesive High-Level Replacement Systems with Negative Application Conditions. Technical report, TU Berlin, 2007.

[LEO06] L. Lambers, H. Ehrig, F. Orejas. Conflict Detection for Graph Transformation with Negative Application Conditions. In *Proceedings of ICGT'06*. LNCS 4178, pp. 61–76. Springer, 2006.

[LEOP08] L. Lambers, H. Ehrig, F. Orejas, U. Prange. Parallelism and Concurrency in Adhesive High-Level Replacement Systems with Negative Application Conditions. *ENTCS*, 2008. to appear.

[LO04] M. Llorens, J. Oliver. Structural and Dynamic Changes in Concurrent Systems: Reconfigurable Petri Nets. *IEEE Transactions on Computers* 53(9):1147–1158, 2004.

[LO06a] M. Llorens, J. Oliver. A Basic Tool for the Modeling of Marked-Controlled Reconfigurable Petri Nets. *ECEASST* 2:13, 2006.

[LO06b] M. Llorens, J. Oliver. Marked-Controlled Reconfigurable Workflow Nets. In *SYNASC*. Pp. 407–413. IEEE Computer Society, 2006.

[MM90] J. Meseguer, U. Montanari. Petri Nets are Monoids. *Information and Computation* 88(2):105–155, 1990.

[PU03] J. Padberg, M. Urbášek. Rule-Based Refinement of Petri Nets: A Survey. In *Petri Net Technology for Communication-Based Systems*. LNCS 2472, pp. 161–196. Springer, 2003.

[Rei08] A. Rein. Reconfigurable Petri Systems with Negative Application Conditions. Technical report 2008/01, TU Berlin, 2008. Diploma Thesis, to appear.

[Roz97] G. Rozenberg (ed.). *Handbook of Graph Grammars and Computing by Graph Transformation, Vol 1: Foundations*. World Scientific, 1997.