Property-Preserving Transformations of Elementary Net Systems Based on Morphisms

Luca Bernardinello[2], Irina Lomazova[1], Roman Nesterov[1,2(✉)], and Lucia Pomello[2]

[1] HSE University, 20 Myasnitskaya Ulitsa, 101000 Moscow, Russia
rnesterov@hse.ru
[2] Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, Viale Sarca 336 - Edificio U14, 20126 Milan, Italy

Abstract. Structural transformations that preserve properties of formal models of concurrent systems make their verification easier. We define structural transformations that allow to abstract and refine elementary net systems. Relations between abstract models and their refinements are formalized using morphisms. Transformations proposed in this paper induce morphisms between elementary net systems as well as preserve their behavioral properties, especially deadlocks. We also show the application of the proposed transformations to the construction of a correct composition of interacting workflow net components.

Keywords: Petri nets · transformations · abstraction · refinement · morphisms

1 Introduction

Petri nets are widely used for modeling concurrent systems as well as for proving their important behavioral properties. Due to the well-known state explosion problem, there are various *structural* techniques developed in Petri net theory. The main advantage of structural techniques is the possibility to verify behavioral properties of Petri nets without computing their reachable markings.

Structural Petri net transformations that preserve classical properties like boundedness, liveness, covering by place invariants make verification of parallel systems easier. On the one hand, starting from a sophisticated model, it is possible to apply *reduction* transformations preserving properties of the initial model and then verify properties using a simplified model. On the other hand, having a simple abstract model, it is possible to apply *refinement* transformations that yield a more detailed model reflecting properties of an initial abstraction.

Petri net transformations have been first described in several works (see, for example, [5,6,14,15,21]), where the authors have defined simple yet powerful local structural reductions and extensions. It has been shown that liveness,

This work is supported by MIUR and the Basic Research Program at HSE University.

M. Koutny et al. (Eds.): ToPNoC XVI, LNCS 13220, pp. 1–23, 2022.
https://doi.org/10.1007/978-3-662-65303-6_1

boundedness (safeness), covering by place invariants, home states and proper termination can be preserved by these transformations.

Free choice Petri nets [11] are also widely adopted to model behavior of parallel systems for their structural constraints on conflicts. The work [9] gives a *complete* set of reduction/synthesis transformations that allows to obtain every live and bounded free choice net. Within the framework of bipolar synchronization schemes [10] strictly related to free choice Petri nets, the authors have also defined a set of local reduction and synthesis rules that yield only well behaved synchronization schemes.

Another series of works [8,18] is devoted to the use of graph transformations in a categorical setting (the double-pushout approach). These transformations are applied to model and analyze behavior of re-configurable systems.

Place [20] and, more generally, resource (sub-marking) [12] bisimulation are also powerful tools to reduce Petri net graphs preserving their observable behavior. These techniques are based on reducing places and resources in Petri nets if they produce bisimilar behavior.

Petri net *morphisms* give a natural yet rigid framework to formalize structural property-preserving relations [7,13,22]). In particular, in [17], morphisms inducing bisimulations have been discussed.

For elementary net systems (EN systems) [3,19] – a basic class of models in net theory – α-morphisms have been introduced in [2]. They help to formalize relations between abstract models and their refinements. Moreover, α-morphisms preserve behavioral properties (reachable markings) as well as reflect them under specific local requirements. However, the direct application of the definition of α-morphisms is rather difficult.

Thus, the main purpose of this paper is to define a set of *local* abstraction/refinement transformations for EN systems. A local transformation acts only on a specific subnet, while the rest of the EN system remains unchanged. We consider EN systems with labeled transitions, where labels specify interactions with the environment. We present transformation rules preserving labeled transitions, while unlabeled transitions are reduced. As a result of applying these transformations, an initial EN system and a transformed EN system are related by an α-morphism, and their reachable markings and, especially, deadlocks are preserved. Interestingly enough, it is also shown that simple Petri net transformations introduced earlier in the literature also yield corresponding α-morphisms.

In addition, we provide two cases of applying transformations defined in our study. Abstraction transformations are used in the context of building a correct composition of interacting workflow nets according to an approach described in [4]. Its correctness is based on abstracting component models with the help of α-morphisms. Refinement transformations are exploited to refine the formal models of abstract interaction patterns [16], which give ready-to-use solutions to organize correct interactions of components in complex parallel systems.

This paper is organized as follows. The following section gives the basic definitions. In Sect. 3, we define local abstraction and refinement transformations for EN systems inducing α-morphisms. Section 4 discusses the application of the proposed transformations to the problem of constructing correct systems with interacting workflow net components, and Sect. 5 concludes the paper.

2 Preliminaries

Here we provide the basic definitions used in the paper.

Let A, B be two sets. A function f from A to B is denoted by $f\colon A \to B$. The set of all finite non-empty sequences over A is denoted by A^+. The set $A^* = A^+ \cup \{\epsilon\}$ is the set of all finite sequences over A, where ϵ is the empty sequence.

A *net* is a triple $N = (P, T, F)$, where P and T are disjoint sets of places and transitions respectively, and $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*. Places are depicted by circles, transitions – by boxes, and the flow relation – by arcs.

Let $N = (P, T, F)$ be a net. The *preset* of $x \in P \cup T$ is the set ${}^\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$. The *postset* of $x \in P \cup T$ is the set $x^\bullet = \{y \in P \cup T \mid (x, y) \in F\}$. The *neighborhood* of $x \in P \cup T$ is the set ${}^\bullet x^\bullet = {}^\bullet x \cup x^\bullet$. N is *P-simple* iff $\forall p_1, p_2 \in P\colon {}^\bullet p_1 = {}^\bullet p_2$ and $p_1{}^\bullet = p_2{}^\bullet$ implies that $p_1 = p_2$. We consider nets without self-loops, i.e., $\forall t \in T\colon {}^\bullet t \cap t^\bullet = \varnothing$, and without isolated transitions, i.e., $\forall t \in T\colon |{}^\bullet t| \geq 1$ and $|t^\bullet| \geq 1$.

Let $N = (P, T, F)$ be a net, and $Y \subseteq P \cup T$. Then ${}^\bullet Y = \bigcup_{y \in Y} {}^\bullet y$, $Y^\bullet = \bigcup_{y \in Y} y^\bullet$, and ${}^\bullet Y^\bullet = {}^\bullet Y \cup Y^\bullet$. $N(Y)$ denotes the subnet of N *generated* by Y, i.e., $N(Y) = (P \cap Y, T \cap Y, F \cap (Y \times Y))$. The set ${}^\circ N(Y) = \{y \in Y \mid \exists z \in (P \cup T) \setminus Y\colon (z, y) \in F \text{ or } {}^\bullet y = \varnothing\}$ is the *input* border, and the set $N(Y)^\circ = \{y \in Y \mid \exists z \in (P \cup T) \setminus Y\colon (y, z) \in F \text{ or } y^\bullet = \varnothing\}$ is the *output* border of $N(Y)$.

A *marking* in a net $N = (P, T, F)$ is a subset of its places $m \subseteq P$. A marking m has a *contact* if $\exists t \in T\colon {}^\bullet t \subseteq m$ and $t^\bullet \cap m \neq \varnothing$. An *elementary net system* (EN system) is a tuple $N = (P, T, F, m_0)$, where $m_0 \subseteq P$ is the *initial* marking. A marking m is shown by putting black dots inside places belonging to m.

A *state machine* is a connected net $N = (P, T, F)$, s.t. $\forall t \in T\colon |{}^\bullet t| = |t^\bullet| = 1$. The subnet of an EN system $N = (P, T, F, m_0)$ generated by $C \subseteq P$ and ${}^\bullet C^\bullet$, i.e., $N(C \cup {}^\bullet C^\bullet)$, is a sequential component of N iff it is a state machine and it has a single token in its initial marking. An EN system $N = (P, T, F, m_0)$ is *covered* by sequential components if every place in N belongs to at least one sequential component. Then N is called state machine decomposable (SMD). For instance, an EN system shown in Fig. 1 has two sequential components generated by $C_1 = \{p_1, p_3, p_4, p_7\}$ and ${}^\bullet C_1{}^\bullet$ as well as by $C_2 = \{p_2, p_5, p_6, p_7\}$ and ${}^\bullet C_2{}^\bullet$. Different sequential components in an SMD-EN system can share both places and transitions. When they share a transition, it is natural to say that sequential components *synchronize*.

The *firing rule* defines the behavior of a net system. A *marking* m in an EN system $N = (P, T, F, m_0)$ *enables* a transition $t \in T$, denoted $m[t\rangle$, iff ${}^\bullet t \subseteq m$ and $t^\bullet \cap m = \varnothing$. When an enabled transition t *fires*, N evolves to a new marking $m' = m \setminus {}^\bullet t \cup t^\bullet$, denoted $m[t\rangle m'$. A sequence $w \in T^*$ is a *firing sequence* of N iff $m_0[t_1\rangle m_1[t_2\rangle \ldots m_{n-1}[t_n\rangle m_n$ and $w = t_1 \ldots t_n$. Then we write $m_0[w\rangle m_n$. The set of all firing sequences of N is denoted by $FS(N)$.

A *marking* m in an EN system $N = (P, T, F, m_0)$ is *reachable* iff $\exists w \in FS(N)\colon m_0[w\rangle m$. The set of all markings reachable from m is denoted by $[m\rangle$. It can be checked that reachable markings in an SMD-EN system are free from contacts. A reachable marking is a *deadlock* if it does not enable any transition.

A deadlock in an SMD-EN system $N = (P, T, F, m_0)$ can be interpreted as a poor synchronization of its sequential components. Since reachable markings in SMD-EN systems are contact-free, we can consider only those deadlocks, which are caused by the absence of tokens in some input places of transitions. For instance, Fig. 1 shows two deadlocks $\{p_3, p_6\}$ and $\{p_4, p_5\}$ that are reachable in the same SMD-EN system from the initial marking $\{p_1, p_2\}$. These deadlocks result from the independent resolution of the local conflicts between t_1 and t_2 as well as t_3 and t_4 by two sequential components: the left generated by $C_1 = \{p_1, p_3, p_4, p_7\}$ and ${}^\bullet C_1{}^\bullet$ and the right generated by $C_2 = \{p_2, p_5, p_6, p_7\}$ and ${}^\bullet C_2{}^\bullet$. In addition, if these local conflicts are resolved differently, s.t. transition t_5 (t_6) is enabled, then it is possible to reach the other deadlock $\{p_7\}$, which can be interpreted as the proper final state of the SMD-EN system from Fig. 1, since $p_7{}^\bullet = \varnothing$.

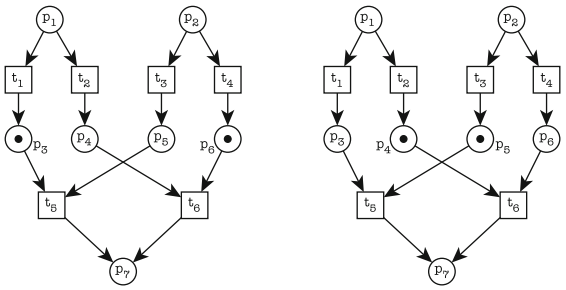


Fig. 1. Two deadlocks after the poor synchronization of sequential components

Abstraction/refinement relations between SMD-EN systems are formalized using α-morphisms introduced in [2]. Below we give the formal definition and briefly discuss the main intuition behind α-morphisms.

Definition 1. *Let $N_i = (P_i, T_i, F_i, m_0^i)$ be an SMD-EN system, $X_i = P_i \cup T_i$ with $i = 1, 2$, where $X_1 \cap X_2 = \varnothing$. An α-morphism from N_1 to N_2 is a total surjective map $\varphi\colon X_1 \to X_2$, also denoted $\varphi\colon N_1 \to N_2$, s.t.:*

1. $\varphi(P_1) = P_2$.
2. $\varphi(m_0^1) = m_0^2$.
3. $\forall t_1 \in T_1\colon$ *if* $\varphi(t_1) \in T_2$, *then* $\varphi({}^\bullet t_1) = {}^\bullet\varphi(t_1)$ *and* $\varphi(t_1{}^\bullet) = \varphi(t_1)^\bullet$.
4. $\forall t_1 \in T_1\colon$ *if* $\varphi(t_1) \in P_2$, *then* $\varphi({}^\bullet t_1{}^\bullet) = \{\varphi(t_1)\}$.
5. $\forall p_2 \in P_2\colon$
   (a) $N_1(\varphi^{-1}(p_2))$ *is an acyclic net.*
   (b) $\forall p_1 \in {}^\bigcirc N_1(\varphi^{-1}(p_2))\colon \varphi({}^\bullet p_1) \subseteq {}^\bullet p_2$ *and if* ${}^\bullet p_2 \neq \varnothing$, *then* ${}^\bullet p_1 \neq \varnothing$.
   (c) $\forall p_1 \in N_1(\varphi^{-1}(p_2))^\bigcirc\colon \varphi(p_1{}^\bullet) = p_2{}^\bullet$.
   (d) $\forall p_1 \in P_1 \cap \varphi^{-1}(p_2)\colon p_1 \notin {}^\bigcirc N_1(\varphi^{-1}(p_2)) \Rightarrow \varphi({}^\bullet p_1) = p_2$ *and* $p_1 \notin N_1(\varphi^{-1}(p_2))^\bigcirc \Rightarrow \varphi(p_1{}^\bullet) = p_2$.
   (e) $\forall p_1 \in P_1 \cap \varphi^{-1}(p_2)\colon$ *there is a sequential component* $N' = (P', T', F')$ *in* N_1, *s.t.* $p_1 \in P'$, $\varphi^{-1}({}^\bullet p_2{}^\bullet) \subseteq T'$.

By definition, an α-morphism allows one to substitute a place in an abstract net system N_2 with an acyclic subnet in N_1. The main motivation behind the use of α-morphisms is the ability to ensure that behavioral properties of an abstract model hold in its refinement.

Let $\varphi\colon N_1 \to N_2$ be an α-morphism. Two main properties of α-morphisms, valid without the additional restrictions, are as follows:

1. A subnet $N_1(\varphi^{-1}(p))$ in N_1, which refines a place p in N_2 behaves exactly as p. Firstly, no tokens are left in the places of $N_1(\varphi^{-1}(p))$ after firing a transition in $N_1(\varphi^{-1}(p))^{\bigcirc}$. Secondly, no transitions in ${}^{\bigcirc}N_1(\varphi^{-1}(p))$ are enabled, when there is a token in the places of $N_1(\varphi^{-1}(p))$.
2. The image of a reachable marking $m \in [m_0^1\rangle$ in N_1 is also a reachable marking in N_2, i.e., $\varphi(m) \in [m_0^2\rangle$. Moreover, $\varphi(m)$ enables the images of transitions enabled by m. In other words, α-morphisms *preserve* reachable markings and transition firings.

The converse of the second property is not always valid, i.e., α-morphisms do not *reflect* reachable markings. As shown in [2], it is necessary to impose additional constraints on the subnets in N_1, which can refine places in N_2.

Figure 2, borrowed from [4], shows an example of α-morphism, where N_1 is a refinement of an abstract SMD-EN system N_2. The subnet $N_1(\varphi^{-1}(p_2))$ in N_1 refines the place p_2 in N_2, and transitions g and h are split into transitions g_1, g_2 and h_1, h_2 in N_1, respectively.

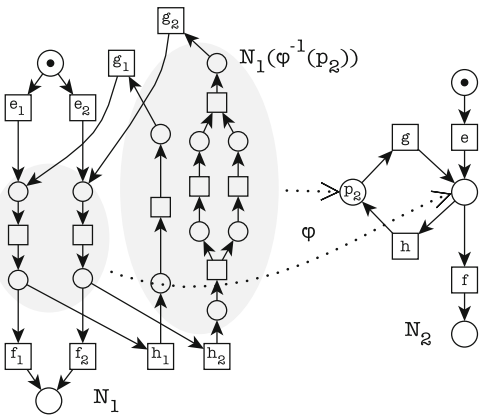


Fig. 2. An α-morphism $\varphi\colon N_1 \to N_2$

3 Structural Transformations of SMD-EN Systems

The direct application of Definition 1 to abstract and refine EN systems may be difficult. In this paper, we consider the use of *structural* transformations of EN systems that induce corresponding α-morphisms. The main purpose of our

study is to develop a system of *local* abstraction/refinement transformations of EN systems and to study properties of these transformations.

It is easy to see that an EN system can have several possible abstractions depending on the detail level. To reduce this ambiguity, we add labels to some transitions in EN-systems. Labeled transitions model actions through which an EN system communicates with its environment. Transformations preserve labeled transitions and minimize the number of local transitions corresponding to the internal behavior of an EN system. Specifically, we plan to apply local transformations to construct a correct composition of synchronously and asynchronously interacting workflow nets as described in [4], where labeled transitions in workflow nets model component synchronizations and message exchange.

Let $N = (P, T, F, m_0)$ be an SMD-EN system, and Λ be an alphabet of communication action names. A transition labeling function is a surjective function $h : T \to \Lambda \cup \{\tau\}$, where $\tau \notin \Lambda$ is the special label of a local action.

We define structural transformation *rules* inducing α-morphisms between initial and transformed EN systems. A transformation rule is a structure $\rho = (L, c_L, R, c_R)$, where:

1. L is the *left* part of a rule that is a subnet in an EN system to be transformed.
2. c_L – flow relation and transition labeling constraints imposed on L.
3. R is the *right* part of a rule that is a subnet replacing L in a net system.
4. c_R – flow relation, marking, transition labeling constraints imposed on R.

L together with c_L define *applicability* constraints of a transformation rule, whereas R and c_R define the transformation itself. We do not give a complete formalization of c_L and c_R, since specific constraints are discussed in the following section. They are necessary to construct an α-morphism between the initial and the transformed EN system. An α-morphism associated with a transformation rule ρ is denoted φ_ρ.

Thus, a transformation rule $\rho = (L, c_L, R, c_R)$ is *applicable* to an SMD-EN system $N = (P, T, F, m_0)$ if there exists a subnet in N isomorphic to L satisfying structural and labeling constraints c_L.

Let $N = (P, T, F, m_0)$ be an SMD-EN net system, and $\rho = (L, c_L, R, c_R)$ be a transformation rule applicable to N. Let $N(X_L)$ be the subnet of N, generated by $X_L \subseteq P \cup T$, s.t. it is isomorphic to L. Then we say that ρ is applicable to the subnet $N(X_L)$ in N. Application of ρ to N includes the following steps:

1. Remove the subnet $N(X_L)$ from N.
2. Add the subnet corresponding to the right part R of ρ to N connecting it with the nodes in the neighborhood ${}^\bullet X_L{}^\bullet$ of the removed subnet.
3. Make necessary changes, i.e., relabel transitions and add tokens to places, in an inserted subnet according to c_R.

The effect of applying ρ to a subnet $N(X_L)$ in N is denoted by $\rho(N, X_L) = (P', T', F', m'_0)$ with a new transition labeling function $h' : T' \to \Lambda \cup \{\tau\}$.

3.1 Abstraction Rules

In this section, we define five simple abstraction rules. They help to abstract SMD-EN systems with labeled transitions. Abstraction rules induce α-morphisms and, correspondingly, preserve reachable markings and deadlocks in EN systems.

For what follows, let $N = (P, T, F, m_0)$ be an SMD-EN system with a transition labeling function $h : T \to \Lambda \cup \{\tau\}$.

A1: Place Simplification

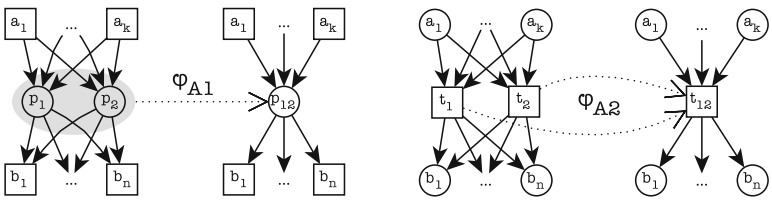
- *applicability constraints*: two places $p_1, p_2 \in P$ in N with the same neighborhood (${}^\bullet p_1 = {}^\bullet p_2$ and $p_1{}^\bullet = p_2{}^\bullet$) as shown in Fig. 3(a).
- *transformation:* fusion of p_1 and p_2 into a single place p_{12}, where ${}^\bullet p_{12} = {}^\bullet p_1 = {}^\bullet p_2$, $p_{12}{}^\bullet = p_1{}^\bullet = p_2{}^\bullet$ and $p_{12} \in m_0' \Leftrightarrow (p_1 \in m_0$ and $p_2 \in m_0)$.
- *α-morphism* $\varphi_{A1} \colon N \to N'$, where $N' = \rho_{A1}(N, \{p_1, p_2\})$, maps places p_1 and p_2 in N to the place p_{12} in N'. For other nodes in N, φ_{A1} is the identity mapping between N and N'.

Place simplification is one of the most basic Petri net transformations. It has been discussed earlier, for instance, in [14] (cf. "fusion of parallel places") and in [5] (cf. "simplification of redundant places").

A2: Transition Simplification

- *applicability constraints*: two transitions $t_1, t_2 \in T$ in N with the same neighborhood and label (${}^\bullet t_1 = {}^\bullet t_2$, $t_1{}^\bullet = t_2{}^\bullet$ and $h(t_1) = h(t_2)$), see Fig. 3(b)).
- *transformation:* fusion of t_1 and t_2 into a single transition t_{12}, where ${}^\bullet t_{12} = {}^\bullet t_1 = {}^\bullet t_2$, $t_{12}{}^\bullet = t_1{}^\bullet = t_2{}^\bullet$ and $h'(t_{12}) = h(t_1) = h(t_2)$.
- *α-morphism* $\varphi_{A2} \colon N \to N'$, where $N' = \rho_{A2}(N, \{t_1, t_2\})$, maps transitions t_1 and t_2 in N to the transition t_{12} in N'. For other nodes in N, φ_{A2} is the identity mapping between N and N'.

Transition simplification (without labeling constraints) is one of the basic Petri net transformations as well. It has been considered, for instance, in [14] (cf. "fusion of parallel transitions").



(a) abstraction rule A1 (b) abstraction rule A2

Fig. 3. Place and transition simplification

A3: Local Transition Elimination

- *applicability constraints:* a transition $t \in T$ in N, s.t. $h(t) = \tau$ and:
  1. ${}^\bullet t = \{p_1\}$ and $t^\bullet = \{p_2\}$;
  2. ${p_1}^\bullet = {}^\bullet p_2 = \{t\}$;
  3. ${}^\bullet p_1 \neq \varnothing$ or ${p_2}^\bullet \neq \varnothing$;
  4. ${}^\bullet p_1 \cap {p_2}^\bullet = \varnothing$.
- *transformation:* fusion of t, p_1 and p_2 into a single place p_{12}, where ${}^\bullet p_{12} = {}^\bullet p_1$, ${p_{12}}^\bullet = {p_2}^\bullet$ and $p_{12} \in m'_0 \Leftrightarrow (p_1 \in m_0$ or $p_2 \in m_0)$.
- *α-morphism* $\varphi_{A3} \colon N \to N'$, where $N' = \rho_{A3}(N, \{p_1, t, p_2\})$, maps t, p_1 and p_2 in N to the place p_{12} in N'. For other nodes in N, φ_{A3} is the identity mapping between N and N'.

Figure 4 shows left and right parts of this rule as well as construction of the α-morphism φ_{A3}. The applicability constraints of ρ_{A3} are aimed to avoid generating isolated places and self-loops in $\rho_{A3}(N, \{p_1, t, p_2\})$. The similar transition transformation "pre-fusion" has been discussed in [5], where it has been expressed as fusion of two transitions connected by a place.

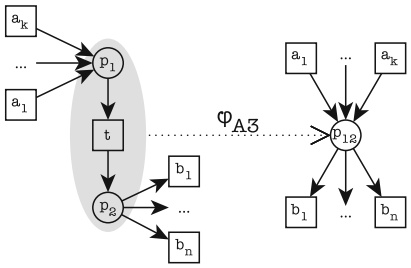


Fig. 4. Abstraction rule A3: local transition elimination

The abstraction rules defined above can be easily generalized: to sets of places and transitions (for ρ_{A1} and ρ_{A2} respectively) or to a "chain" of local transitions (for ρ_{A3}). We propose to apply a simple abstraction rule several times rather than to complicate their applicability constraints.

A4: Postset-Empty Place Simplification

- *applicability constraints:* two places p_1 and p_2 in N, s.t. ${p_1}^\bullet = {p_2}^\bullet = \varnothing$ and:
  1. ${}^\bullet p_1 \cap {}^\bullet p_2 = \varnothing$;
  2. $\forall C \subseteq P$: if $N(C \cup {}^\bullet C^\bullet)$ is a sequential component, then $p_1 \in C \Leftrightarrow p_2 \in C$.
- *transformation:* fusion of p_1 and p_2 into a single place p_{12}, s.t. ${}^\bullet p_{12} = {}^\bullet p_1 \cup {}^\bullet p_2$, ${p_{12}}^\bullet = {p_1}^\bullet = {p_2}^\bullet$ and $p_{12} \in m'_0 \Leftrightarrow (p_1 \in m_0$ or $p_2 \in m_0)$, see Fig. 5.
- *α-morphism* $\varphi_{A4} \colon N \to N'$, where $N' = \rho_{A4}(N, \{p_1, p_2\})$, maps p_1 and p_2 in N to the place p_{12} in N'. For other nodes in N, φ_{A4} is the identity mapping between N and N'.

It is necessary to check that there are no sequential components distinguishing p_1 and p_2 in order to preserve state machine decomposability after transformation. Therefore we also satisfy the requirement 5e of Definition 1.

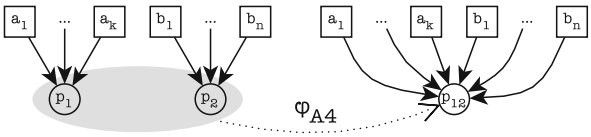


Fig. 5. Abstraction rule A4: Postset-empty place simplification

A5: Preset-Disjoint Transition Simplification

In this abstraction rule ρ_{A5}, we fuse two transitions that have the same postset and disjoint presets as opposed to the abstraction rule ρ_{A2}. Applicability constraints of this rule do not allow us to lose deadlocks present in an initial EN system by abstracting it. The problem of losing deadlocks is the consequence of the fact that α-morphisms do not *reflect* reachable markings without additional restrictions, as discussed in Sect. 2. In the setting of our study, this means that an inverse image of a reachable marking that enables transitions in an abstract model may be a deadlock in an initial EN system.

Let us illustrate the problem of losing deadlocks by the following example based on the EN system shown in Fig. 1. Recall that it has two deadlocks $\{p_3, p_6\}$ and $\{p_4, p_5\}$ reachable from the initial marking $\{p_1, p_2\}$. These deadlocks are caused by the fact that conflicts are resolved independently by two sequential components. Suppose that the two transitions t_5 and t_6 have the same label λ. Then, according to Definition 1, it is possible to fuse p_3 with p_4, p_5 with p_6 and t_5 with t_6 correspondingly (see Fig. 6, where the fusion is indicated by the indices, and the α-morphism φ is provided as well). The image t_{56} of t_5 and t_6 has two places in its preset, and there exists reachable marking $\{p_{34}, p_{56}\}$ enabling t_{56}. However, there exists an inverse image of the marking $\{p_{34}, p_{56}\}$, e.g., the deadlock $\{p_3, p_6\}$ that does not enable an inverse image of t_{56}.

Thus, we impose additional constraints on places in the presets of two transitions to be fused in such a way that if there is a deadlock containing places in the presets of both transitions, then it should not be possible to fuse these transitions. Preset-disjoint transition simplification is defined as follows:

– *applicability constraints:* two transitions t_1 and t_2, s.t. $h(t_1) = h(t_2)$ and:
  1. ${}^\bullet t_1 \cap {}^\bullet t_2 = \varnothing$ and $|{}^\bullet t_1| = |{}^\bullet t_2|$;
  2. $t_1{}^\bullet = t_2{}^\bullet$;
  3. $\forall a \in {}^\bullet t_1 \, \forall b \in {}^\bullet t_2 \, \exists C \subseteq P$: $a, b \in C$ and $N(C \cup {}^\bullet C^\bullet)$ is a sequential component.

– *transformation:* fusion of t_1 and t_2 into a single transition t_{12} with $h'(t_{12}) = h(t_1) = h(t_2)$, ${t_{12}}^\bullet = {t_1}^\bullet = {t_2}^\bullet$ and ${}^\bullet t_{12} = \{(a,b) \mid a \in {}^\bullet t_1, b \in {}^\bullet t_2, g(a) = b\}$, where $g\colon {}^\bullet t_1 \to {}^\bullet t_2$ is a bijection. Input transitions of ${}^\bullet t_1$ and ${}^\bullet t_2$ are preserved, i.e., $\forall (a,b) \in {}^\bullet t_{12}\colon {}^\bullet(a,b) = {}^\bullet a \cup {}^\bullet b$. As for the initial marking m'_0 in $\rho_{A5}(N, \{t_1, t_2\})$, we have $\forall (a,b) \in {}^\bullet t_{12}\colon (a,b) \in m'_0 \Leftrightarrow (a \in m_0 \text{ or } b \in m_0)$.
– *α-morphism* $\varphi_{A5}\colon N \to N'$, where $N' = \rho_{A5}(N, \{t_1, t_2\})$, maps transitions t_1 and t_2 to the transition t_{12} in N' as well as every pair of places $a \in {}^\bullet t_1$ and $b \in {}^\bullet t_2$, where $g(a) = b$, is mapped to the place $(a,b) \in {}^\bullet t_{12}$. For other nodes in N, φ_{A5} is the identity mapping.

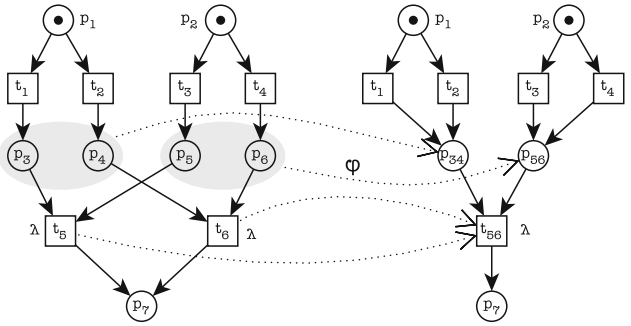


Fig. 6. Deadlocks are not preserved by α-morphisms

In Fig. 7, we provide the left and right parts of the abstraction rule ρ_{A5}, where the pairwise fusion of places is shown only for places a_1 and b_1 with $g(a_1) = b_1$, which are fused into the place $f_1 = (a_1, b_1)$. For other pairs of places, this fusion is performed similarly. The bijection $g\colon {}^\bullet t_1 \to {}^\bullet t_2$ is an integral part of ρ_{A5}, which makes the preset-disjoint transition simplification unambiguous.

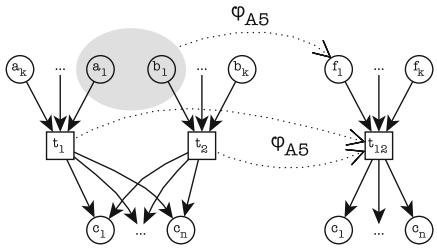


Fig. 7. Abstraction rule A5: Preset-disjoint transition simplification

The third applicability constraint of ρ_{A5} makes sure that every place in ${}^\bullet t_1$ is in conflict with every place in ${}^\bullet t_2$. Then it is easy to check that if there is a

reachable marking in N with a token in ${}^\bullet t_1$, then there cannot be a token in ${}^\bullet t_2$ at the same time. The application of this rule involves pairwise place fusion in ${}^\bullet t_1$ and ${}^\bullet t_2$. According to the requirement on sequential components, we define a bijection $g : {}^\bullet t_1 \to {}^\bullet t_2$ and fuse places in ${}^\bullet t_1$ and ${}^\bullet t_2$ corresponding by g.

Let us consider two more detailed examples of applying the abstraction rule ρ_{A5}. There are two SMD-EN systems N_1 and N_2 shown in Fig. 8. Transitions t_1 and t_2 in N_1 as well as transitions e_1 and e_2 in N_2 are candidates to be fused, since they have the same label λ, share the same postset, whereas their presets are disjoint. We have to check whether places in the presets of these transitions are connected by sequential components. The results of this verification for N_1 and N_2 are given in Table 1, where we provide only sets of places corresponding to sequential components.

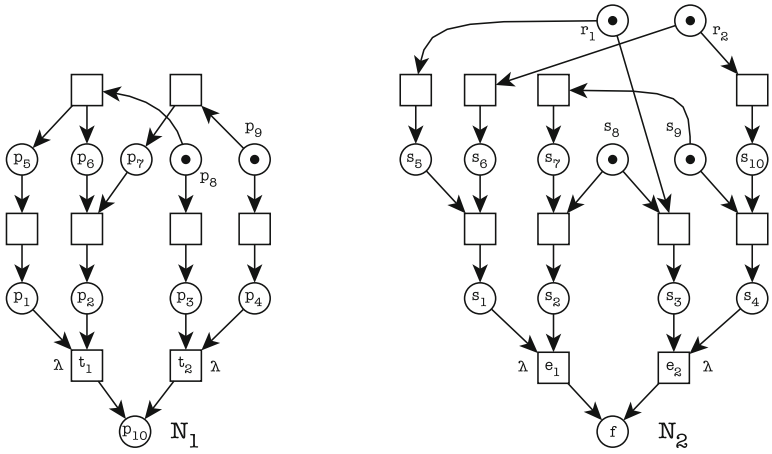


Fig. 8. Two EN systems to check applicability constraints of ρ_{A5}

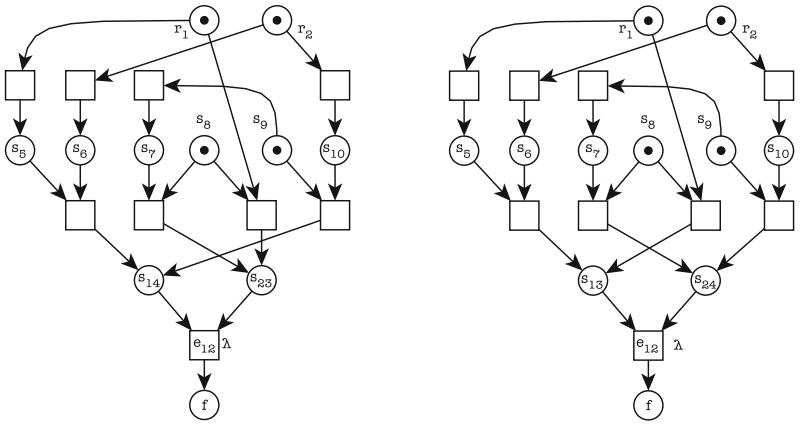
In N_1, there is no sequential component containing places p_1 and p_4. Indeed, there is the deadlock $\{p_1, p_6, p_4\}$ containing places both from ${}^\bullet t_1$ and ${}^\bullet t_2$. Thus, transitions t_1 and t_2 in N_1 cannot be fused without losing this deadlock.

In N_2, we have found sequential components for all pairs of places from ${}^\bullet e_1$ and ${}^\bullet e_2$. Thus, we can fuse these transition according to the abstraction rule ρ_{A5}. There can be two possible transformations depending on the choice of places to be fused, i.e., either s_1 is fused with s_4 (see Fig. 9(a)) or s_1 is fused with s_3 (see Fig. 9(b)). It is enough to choose a single pair of places to be fused, and other pairs of places are dex termined in the only possible way.

It is worth mentioning that application of rule ρ_{A5} can also be straightforwardly extended to the case when transitions t_1 and t_2 have shared places in their presets. In this context, shared places will be preserved by a transformation.

Table 1. Verification of sequential components in N_1 and N_2 from Fig. 8

Sequential components in N_1		Sequential components in N_2	
p_1 and p_3	$\{p_8, p_5, p_3, p_1, p_{10}\}$	s_1 and s_3	$\{r_1, s_5, s_1, s_3, f\}$
p_1 and p_4	NO	s_1 and s_4	$\{r_2, s_6, s_{10}, s_1, s_4, s_{13}\}$
p_2 and p_3	$\{p_8, p_6, p_2, p_3, p_{10}\}$	s_2 and s_3	$\{s_8, s_2, s_3, f\}$
p_2 and p_4	$\{p_9, p_7, p_2, p_4, p_{10}\}$	s_2 and s_4	$\{s_9, s_7, s_2, s_2, f\}$



(a) s_1 is merged with s_4 (b) s_1 is merged with s_3

Fig. 9. Two results of applying the rule ρ_{A5} to N_2 from Fig. 8

3.2 Properties of the Abstraction Rules

We next discuss the main properties of the simple abstraction rules. We denote the set of abstraction rules by $AR = \{\rho_{A1}, ..., \rho_{A5}\}$.

By construction, the application of an abstraction rule induces an α-morphism from the initial SMD-EN system towards the transformed one.

Proposition 1. *Let $\rho \in AR$, s.t. ρ is applicable to a subnet $N(X_L)$ in N. Then there is an α-morphism $\varphi_\rho \colon N \to \rho(N, X_L)$.*

Corollary 1. *Let $\rho_1, \rho_2 \in AR$, s.t. ρ_2 is applicable to a subnet in $\rho_1(N, X_L)$ generated by X'_L. Then there is an α-morphism $\varphi_{\rho_2} \circ \varphi_{\rho_1} : N \to \rho_2(\rho_1(N, X_L), X'_L)$.*

The important property is whether the order of applying abstraction rules matters when at least two abstraction rules are applicable to the same net system. In this case, we distinguish when two abstraction rules (applicable to the same net system) coincide or differ.

Proposition 2. *Let $\rho_1, \rho_2 \in AR$, s.t. ρ_1 is applicable to a subnet $N(X^1_L)$ in N, ρ_2 is applicable to a subnet $N(X^2_L)$ in N and $X^1_L \neq X^2_L$. Then:*

1. *If* $\rho_1 = \rho_2$*, then the effect of applying* ρ_2 *to* $\rho_1(N, X_L^1)$ *is isomorphic to the effect of applying* ρ_1 *to* $\rho_2(N, X_L^2)$.
2. *If* $\rho_1 \neq \rho_2$ *and* $X_L^1 \cap X_L^2 = \varnothing$*, then* $\rho_2(\rho_1(N, X_L^1), X_L^2) = \rho_1(\rho_2(N, X_L^2), X_L^1)$.

The second part of Proposition 2 is easy to check, i.e., the order of applying abstraction rules transforming disjoint subnets is immaterial. However, the first part of this Proposition requires an additional clarification, when $\rho_1 = \rho_2 = \rho_{A5}$. The result of applying the other abstraction rules fully depends on the subnets corresponding to their left parts, which are fixed in Proposition 2.

As discussed above, the bijection g between the input places of two transitions is an integral part of ρ_{A5}. Then we require that for the repeated application of ρ_{A5}, one fix the bijections at the time of checking the applicability constraints. The following example (see Fig. 10) shows a case when ρ_{A5} can be applied twice and explains how to define the correct bijections between input places.

Suppose an SMD-EN system has a subnet shown in Fig. 10(a), which satisfies the applicability constraints of ρ_{A5} for fusing t_1 with t_2, t_2 with t_3, or t_1 with t_3. We need to define two bijections between the input places of any two pairs of transitions, and the bijection for third pair of transitions will be obtained transitively. For instance, let $g_1 \colon {}^\bullet t_1 \to {}^\bullet t_2$ and $g_2 \colon {}^\bullet t_1 \to {}^\bullet t_3$, s.t. $g_1(a_1) = b_1$, $g_1(a_2) = b_2$, $g_2(a_1) = c_2$, and $g_2(a_2) = c_1$. These correspondences between the input places are also shown by dotted lines in Fig. 10(a). Then the third bijection $g_3 \colon {}^\bullet t_2 \to {}^\bullet t_3$ is defined as follows: $g_3(b_2) = c_1$ and $g_3(b_1) = c_2$. Arbitrary definition of the third bijection might break transitivity and, thus, disable the repeated application of the abstraction rule ρ_{A5}. In other words, the number of required bijections corresponds to the number of times ρ_{A5} will be applied.

We next demonstrate that the order of fusing transitions is not important, since the results are isomorphic. Suppose that, firstly, transitions t_1 and t_2 are to be fused. Then they are transformed into a single transition t_{12}, s.t. ${}^\bullet t_{12} = \{(a_1, b_1), (a_2, b_2)\}$ according to g_1. The fusion of t_{12} with t_3 will yield a transition t_{123} with ${}^\bullet t_{123} = \{(a_1, b_1, c_2), (a_2, b_2, c_1)\}$, as shown in Fig. 10(b). Changing the order of the consecutive fusions, we may, for example, obtain a transition t_{231} with ${}^\bullet t_{231} = \{(b_2, c_1, a_2), (b_1, c_2, a_1)\}$ (see Fig. 10(c)), which is isomorphic to the earlier constructed result.

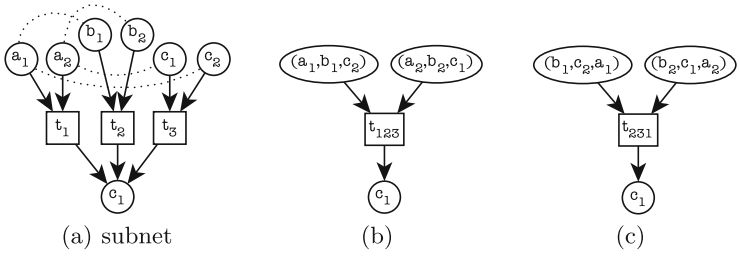


Fig. 10. Repeated application of the abstraction rule ρ_{A5}

Thus, the unambiguity of the repeated application of ρ_{A5} requires that the corresponding bijections between the input places of transition pairs are defined for an initial SMD-EN system.

According to the structural requirements of abstraction rules, we also conclude that if there is a deadlock in an initial net system, then the image of this deadlock is also a deadlock in a transformed net system (see Proposition 3). In proving this statement, we also rely on the fact that α-morphisms *preserve* reachable markings and transition firings, i.e., an image of a reachable marking in a refined EN system is also a reachable marking which, moreover, enables any image of enabled transitions in a refined model.

Proposition 3. *Let $\rho \in AR$, s.t. ρ is applicable to a subnet $N(X_L)$ in N. Let $m \in [m_0\rangle$ be a deadlock in N. Then $\varphi_\rho(m)$ is a deadlock in $\rho(N, X_L)$.*

Proof. Let $N' = \rho(N, X_L)$. If $m^\bullet = \varnothing$, then, by Definition 1, $\varphi_\rho(m)^\bullet = \varnothing$. Thus, $\varphi_\rho(m)$ is a deadlock in N'. If $\exists t \in T\colon {}^\bullet t \cap m \neq \varnothing$, then the proof is done by contradiction. Suppose that $\varphi_\rho(m)$ is not a deadlock. Then either ${}^\bullet\varphi_\rho(t) = \varphi_\rho(m)$, i.e., a transition t and ${}^\bullet t^\bullet$ is mapped to the same place, or ${}^\bullet\varphi_\rho(t) \subseteq \varphi_\rho(m)$, i.e., a marking $\varphi_\rho(m)$ enables $\varphi_\rho(t)$ in N'. A transition t cannot be mapped to a place by φ_ρ since $|{}^\bullet t| > 1$, because there are places in ${}^\bullet t$, s.t. ${}^\bullet t \cap m \neq \varnothing$ and there is at least one place $p \in {}^\bullet t$, s.t. $p \notin m$. If a marking $\varphi_\rho(m)$ enables $\varphi_\rho(t)$ in N', then t is fused with another transition t' by ρ, s.t. ${}^\bullet t' \cap m \neq \varnothing$. This fusion is not allowed by the abstraction rule ρ_{A5}, then there is a contradiction.

3.3 Refinement Rules

In this section, we define four simple refinement rules. They allow one to refine a given SMD-EN system. Three out of four proposed refinement rules are the inverse of abstraction rules discussed in the previous section. Refinement rules also induce α-morphisms. The main difference here is that the direction of α-morphisms is the opposite to the direction of a transformation.

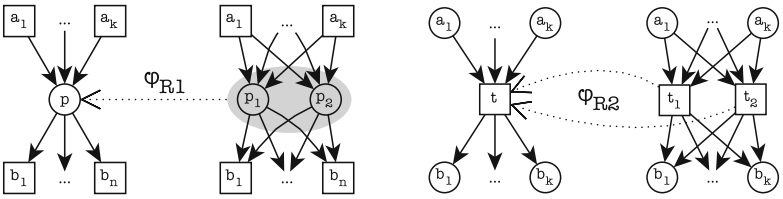
For what follows, let $N = (P, T, F, m_0)$ be an SMD-EN system with a transition labeling function $h : T \to \Lambda \cup \{\tau\}$. Recall also that the effect of applying a transformation rule ρ to N is denoted by $\rho(N, X_L) = (P', T', F', m'_0)$ with a new transition labeling function $h'\colon T' \to \Lambda \cup \{\tau\}$, where $X_L \subseteq P \cup T$ and $N(X_L)$ is the subnet in N transformed by ρ.

R1: Place Duplication

- *applicability constraints:* a place p in N.
- *transformation:* split p into two places p_1 and p_2, where ${}^\bullet p_1 = {}^\bullet p_2 = {}^\bullet p$, $p_1{}^\bullet = p_2{}^\bullet = p^\bullet$ and $(p_1 \in m'_0$ and $p_2 \in m'_0) \Leftrightarrow p \in m_0$ (see Fig. 11(a)).
- *α-morphism* $\varphi_{R1}\colon N' \to N$, where $N' = \rho_{R1}(N, \{p\})$, maps places p_1 and p_2 in N' to the place p in N. For other nodes in N', φ_{R1} is the identity mapping between N' and N.

R2: Transition Duplication

- *applicability constraints:* a transition t in N.
- *transformation:* split t into two transitions t_1 and t_2, where $h'(t_1) = h'(t_2) = h(t)$, ${}^\bullet t_1 = {}^\bullet t_2 = {}^\bullet t$ and ${t_1}^\bullet = {t_2}^\bullet = t^\bullet$ (see Fig. 11(b)).
- *α-morphism* $\varphi_{R2}\colon N' \to N$, where $N' = \rho_{R2}(N, \{t\})$, maps transitions t_1 and t_2 in N' to the transition t in N. For other nodes in N', φ_{R2} is the identity mapping between N' and N.



(a) refinement rule R1 (b) refinement rule R2

Fig. 11. Place and transition duplication

R3: Local Transition Introduction

- *applicability constraints:* a place p in N.
- *transformation:* substitution of p with a transition t and two places p_1, p_2 (see Fig. 12), where:
  1. $h'(t) = \tau$;
  2. ${}^\bullet t = \{p_1\}$ and $t^\bullet = \{p_2\}$;
  3. ${p_1}^\bullet = {}^\bullet p_2 = \{t\}$;
  4. ${}^\bullet p_1 = {}^\bullet p$ and ${p_2}^\bullet = p^\bullet$;
  5. $p \in m_0 \Leftrightarrow ((p_1 \in m'_0$ and $p_2 \notin m'_0)$ or $(p_1 \notin m'_0$ and $p_2 \in m'_0))$.
- *α-morphism* $\varphi_{R3}\colon N' \to N$, where $N' = \rho_3(N, \{p\})$, maps t, p_1 and p_2 in N' to the place p in N. For other nodes in N', φ_{R3} is the identity mapping between N' and N.

Refinement rule ρ_{R1} (ρ_{R2}) can be generalized to the case when a place (a transition) in the initial EN system is split into a set of places (transitions). Refinement rule ρ_{R3} can be generalized to the case when a places in the in initial EN system is replaced with a "chain" of local transitions. These extensions are similar to the possible generalizations of abstraction rules ρ_{A1}, ρ_{A2} and ρ_{A3} discussed above.

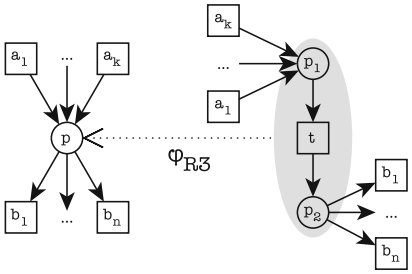


Fig. 12. Refinement rule R3: local transition introduction

R4: Place Split

- *applicability constraints:* a place p in N, s.t. $|{}^\bullet p| > 1$.
- *transformation:* split p into two places p_1 and p_2 (see Fig. 13), where:
  1. ${}^\bullet p_1 \neq \varnothing$ and ${}^\bullet p_2 \neq \varnothing$;
  2. ${}^\bullet p_1 \subset {}^\bullet p$ and ${}^\bullet p_2 \subset {}^\bullet p$;
  3. ${}^\bullet p_1 \cap {}^\bullet p_2 = \varnothing$ and ${}^\bullet p_1 \cup {}^\bullet p_2 = {}^\bullet p$;
  4. $|p_1{}^\bullet| = |p_2{}^\bullet| = |p^\bullet|$, and there is a bijection $f_i \colon p_i{}^\bullet \to p^\bullet$ with $i = 1, 2$, s.t. $\forall x \in p_i{}^\bullet \colon h'(x) = h(f_i(x))$;
  5. $(p_i{}^\bullet)^\bullet = (p^\bullet)^\bullet$ with $i = 1, 2$;
  6. ${}^\bullet(p_i{}^\bullet) \setminus \{p_i\} = {}^\bullet(p^\bullet) \setminus \{p\}$ with $i = 1, 2$;
  7. if $p \in m_0$, then $p_1 \in m_0' \Leftrightarrow p_2 \notin m_0'$.
- *α-morphism* $\varphi_{R4} \colon N' \to N$, where $N' = \rho_{R4}(N, \{p\})$, maps places p_1 and p_2 in N' to to the place p in N and maps each transition $t' \in p_i{}^\bullet$ in N' to a transition $t \in p^\bullet$ in N if $f_i(t') = t$ with $i = 1, 2$. For other nodes in N', φ_{R4} is the identity mapping between N' and N.

While splitting a place p in N, its neighborhood is also split between p_1 and p_2 in $\rho_{R4}(N, \{p\})$. According to constraints 1, 2 and 3, the preset of p is divided into two disjoint, proper and non-empty subsets. According to constraint 4, the postsets of p_1 and p_2 are exactly two copies of the postset of p, s.t. labels of transitions are also preserved. Moreover, by constraints 5 and 6, the input and output places in $p_1{}^\bullet$ and $p_2{}^\bullet$ are the same as the input and output places of $p^\bullet$. These requirements on splitting the neighborhood of p in N are based on the requirements 5b and 5c of Definition 1.

Figure 13 provides the left and right parts of the refinement rule ρ_{R4}, where the corresponding α-morphism maps p_1 and p_2 in the transformed EN system to a place p in the initial EN system. The map from the postsets of p_1 and p_2 to the postset of p is shown only for two pairs of transitions: c_1 is mapped to b_1 since $f_1(c_1) = b_1$; d_1 is also mapped to b_1 since $f_2(d_1) = b_1$, where the bijections f_1 and f_2 are constructed according to constraint 4 of this rule.

3.4 Properties of the Refinement Rules

We continue by discussing the main properties of the proposed refinement rules. Let $RR = \{\rho_{R1}, \ldots, \rho_{R4}\}$ be the set of refinement rules.

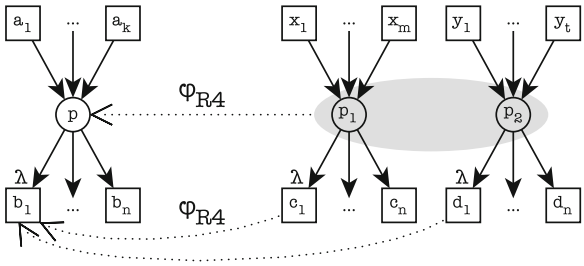


Fig. 13. Refinement rule R4: place split

By construction, the application of a refinement rule induces an α-morphism from a transformed SMD-EN system to an initial SMD-EN system. This also follows from the fact that rules ρ_{R1}, ρ_{R2} and ρ_{R3} are the inverse of the abstraction rules ρ_{A1}, ρ_{A2} and ρ_{A3} respectively.

Proposition 4. *Let $\rho \in RR$, s.t. ρ is applicable to a subnet $N(X_L)$ in N. Then there is an α-morphism $\varphi_\rho \colon \rho(N, X_L) \to N$.*

Corollary 2. *Let $\rho_1, \rho_2 \in RR$, s.t. ρ_2 is applicable to a subnet in $\rho_1(N, X_L)$ generated by X'_L. Then there is an α-morphism $\varphi_{\rho_2} \circ \varphi_{\rho_1} \colon \rho_2(\rho_1(N, X_L), X'_L) \to N$.*

Similarly to the abstraction rules, we also observe that application of the refinement rules does not introduce "new" deadlocks to transformed models, i.e., an image of a deadlock in a refined EN system is also a deadlock already present in an initial abstract EN system.

Proposition 5. *Let $\rho \in RR$, s.t. ρ is applicable to a subnet $N(X_L)$ in N. Let $m' \in [m'_0\rangle$ be a deadlock in $\rho(N, X_L)$. Then $\varphi_\rho(m')$ is a deadlock in N.*

Proof. The proof follows from two facts. Firstly, as discussed in Sect. 2, a deadlock m in an SMD-EN system $N = (P, T, F, m_0)$ is such a reachable marking, where for any transition $t \in T$, s.t. ${}^\bullet t \cap m \neq \varnothing$, there is at least one place $p \in {}^\bullet t$, s.t. $p \notin m$. Secondly, the application of the refinement rules, which result in splitting places (thus, generating new inverse images of reachable markings in an initial EN-system N), fully preserves their neighborhoods.

4 Use of Transformations for Workflow Net Composition

Here we show the application of transformations defined in Sect. 3 to a correct composition of interacting workflow nets. Workflow nets have both initial and final markings. We follow an approach to a composition of *generalized workflow nets* (GWF-nets) described in [4]. The correctness of this approach is achieved through the use of α-morphisms. GWF-nets interact by synchronizations and by sending/receiving messages through asynchronous channels. GWF-net interactions are specified using transition labels. Below we recall main definitions.

In our paper, we consider workflow nets covered by sequential components. As mentioned in [1], state machine decomposability is a basic feature that bridges structural and behavioral properties of workflow nets.

Definition 2. *A generalized workflow net (GWF-net) is an SMD-EN system* $N = (P, T, F, m_0, m_f)$*, where:*

1. ${}^\bullet m_0 = \varnothing$.
2. $m_f \subseteq P$, *s.t.* $m_f \neq \varnothing$ *and* $m_f{}^\bullet = \varnothing$.
3. $\forall x \in P \cup T \, \exists s \in m_0 \, \exists f \in m_f \colon (s, x) \in F^*$ *and* $(x, f) \in F^*$, *where* F^* *is the reflexive transitive closure of* F.

Soundness is the main correctness property of workflow nets.

Definition 3. *A GWF-net* $N = (P, T, F, m_0, m_f)$ *is sound iff:*

1. $\forall m \in [m_0\rangle \colon m_f \in [m\rangle$.
2. $\forall m \in [m_0\rangle \colon m \supseteq m_f \Rightarrow m = m_f$.
3. $\forall t \in T \, \exists m \in [m_0\rangle \colon m[t\rangle$.

Two kinds of transition labels for synchronous and asynchronous interactions are assigned to certain transitions in a GWF-net. The composition of two transition-labeled GWF-nets N_1 and N_2 is also a transition-labeled GWF-net denoted by $N_1 \circledast N_2$, and it is fully defined according to transition labels in N_1 and N_2: (a) fusion of transitions with a common synchronous label, (b) addition of a place for an asynchronous channel between two transitions with complement asynchronous labels.

Figure 14 shows an example of adding a channel represented by a labeled place a that is shown by a smaller circle. It connects transition t_1 in N_1 (label $a!$ corresponds to sending a message to channel a) to transition t_2 in N_2 (label $a?$ corresponds to receiving a message from channel a).

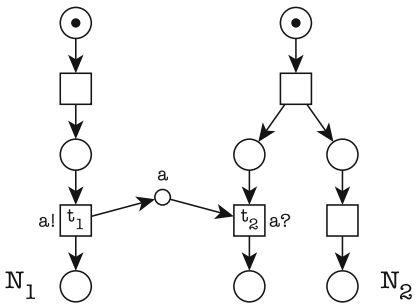


Fig. 14. Addition of a place for an asynchronous channel between two GWF-nets

The main result of [4] on the GWF-net composition correctness is formulated in the following proposition.

Proposition 6 ([4]). *Let R_1, R_2 and A_1, A_2 be four sound transition-labeled GWF-nets, s.t. $\varphi_i\colon R_i \to A_i$ is an α-morphism with $i = 1, 2$. If $A_1 \circledast A_2$ is sound, then $R_1 \circledast R_2$ is sound.*

Thus, the composition of two *detailed* transition-labeled GWF-nets $R_1 \circledast R_2$ is sound if the composition of their *abstractions* $A_1 \circledast A_2$ is sound. Intuitively, $A_1 \circledast A_2$ models an abstract interaction protocol (also referred to as an *interaction pattern*) between detailed transition-labeled GWF-net components. We use transformation rules to define corresponding α-morphisms, as shown further.

4.1 Abstraction of Interacting Workflow Net Components

Here we show the application of the abstraction rules to build abstract representations of interacting transition-labeled GWF-nets. For example, we aim to construct the α-morphism shown in Fig. 2 step by step. Assume that transitions $e_1, e_2, f_1, f_2, g_1, g_2, h_1, h_2$ in N_1 are labeled by names of communication actions from Λ, s.t. $h(e_1) = h(e_2)$, $h(f_1) = h(f_2)$, $h(g_1) = h(g_2)$ and $h(h_1) = h(h_2)$, whereas transitions $y_1, \ldots, y_7$ in N_1 are local, i.e., they are labeled by τ.

Firstly, local transitions $y_1, \ldots, y_5$ can be eliminated using rule ρ_{A3} five times. After collapsing these local transitions, we simplify places p_1 and p_2 (by rule ρ_{A1}) that are generated from eliminating local transitions y_4 and y_5 correspondingly. Figure 15 gives a concise illustration of these transformations.

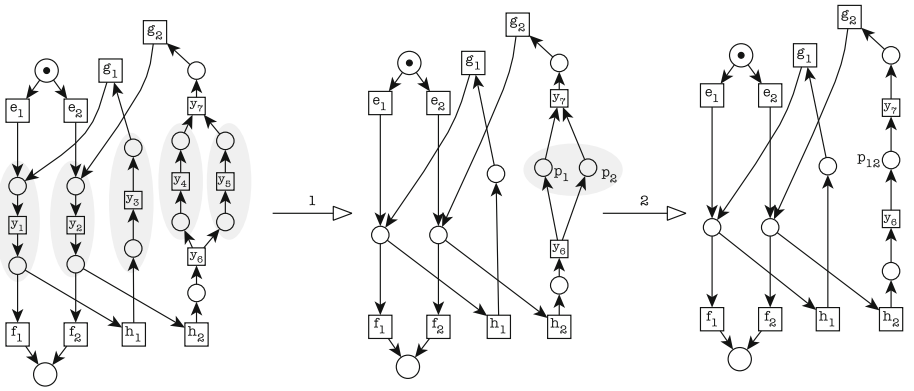


Fig. 15. Abstracting a GWF-net: steps 1 and 2

Now local transitions y_7 and y_8 are also eliminated using rule ρ_{A3} twice (see transformation 3 in Fig. 16). Unfortunately, the fourth transformation shown in Fig. 16 cannot be obtained using the existing simple abstraction rules. We fuse transitions f_1 and f_2, h_1 and h_2, g_1 and g_2 preserving their labels as well as

we fuse places p_3 and p_4, p_5 and p_6 in the neighborhood of these transitions. Intuitively, this may be seen as an example of a direct application of Definition 1 by constructing appropriate fusions. We plan to investigate possible local transformations applicable in this case in the future.

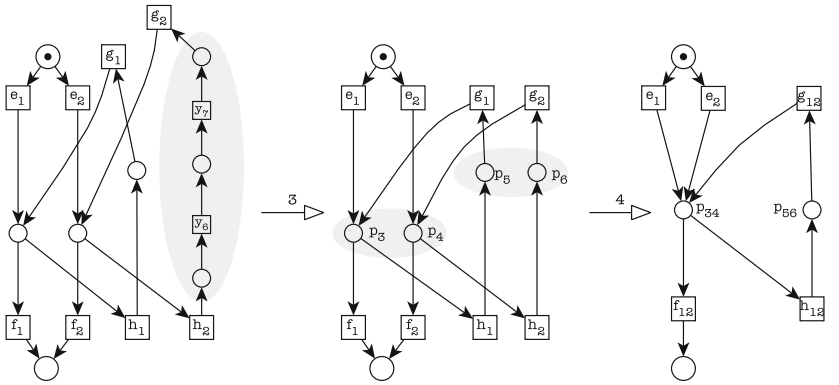


Fig. 16. Abstracting a GWF-net: steps 3 and 4

Finally, we simplify transitions e_1 and e_2 (by rule ρ_{A2}) and obtain the target abstract EN system N_2 previously demonstrated in Fig. 2. To construct the corresponding map between models, we need to compose all α-morphisms induced by applied abstraction rules and by a direct application of Definition 1.

4.2 Refinement of Interaction Patterns

In this section, we apply the refinement rules to solve the inverse problem: the construction of a detailed system model that preserves properties of an initial abstract model. As described above, a protocol of interactions between transition-labeled GWF-net components is represented by a composition of their abstractions. In [16], typical interaction patterns for asynchronously communicating transition-labeled GWF-nets have been discussed. Abstract interaction patterns provide generic solutions that can be used to model and verify component interactions in large-scale distributed systems.

Consider a pattern shown in Fig. 17 (refer to "Send-Receive" interaction pattern in [16]). It models a message exchange between two components: the left sends a message to the right, while the right sends a response back to the left. We aim to construct a possible refinement of this pattern using our rules.

Figure 18 gives a concise illustration for building a possible refinement of the "Send-Receive" interaction pattern. Firstly, we duplicate place p_1 (rule ρ_{R1}) and introduce a local transition instead of place p_2 (rule ρ_{R3}). Secondly, we also introduce local transitions instead of the copies of place p_1, and we duplicate transition t_2 (rule ρ_{R2}). The last transformation shown in Fig. 18 is the split of place p_3 according to the refinement rule ρ_{R4}. Refinement process may be continued until the target detail level is achieved.

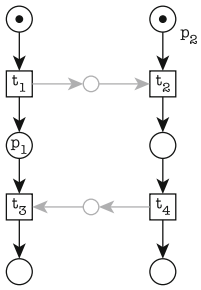


Fig. 17. "Send-Receive" asynchronous interaction pattern

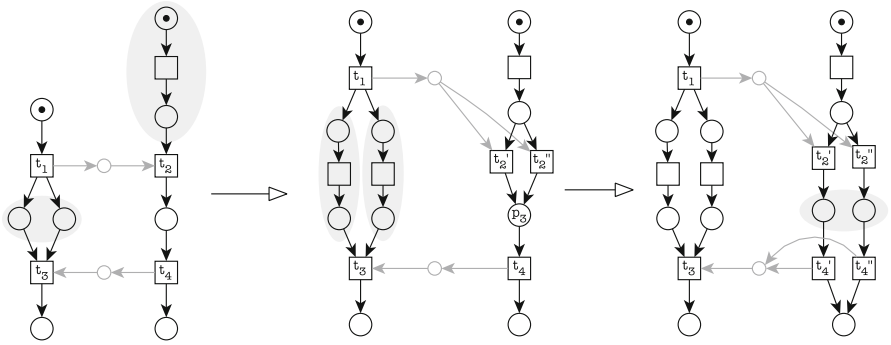


Fig. 18. Refinement of the interaction pattern from Fig. 17

5 Conclusion

In this paper, we have studied the problem of abstracting and refining elementary net systems with the help of α-morphisms. Direct construction of these morphisms is a complicated task. To solve it, we have proposed a set of abstraction/refinement transformation rules. The step-wise application of these transformation rules induces corresponding α-morphisms between initial and transformed models. Some of the transformations (the abstraction rules A1–A3 and their refinement counterparts) have been already discussed in the literature, while the others are new and have been developed in accordance with the definition of α-morphisms. We note that structural applicability constraints of the proposed transformation rules can be efficiently computed. Moreover, locality of abstraction/refinement transformation rules proposed in our study allows us to preserve and reflect not only reachable markings, but also deadlocks. Thus, structural constraints of transformation rules make impossible to lose or introduce deadlocks in models. In addition, we have demonstrated how transformation rules can be applied to compose models of interacting workflow net components.

There are several open theoretical questions that we intend to study in future. It is planned to extend transformations defined in the paper with more liberal

yet controlled ways of introducing concurrency rather than by duplicating places only, e.g., it is possible to consider introduction and detection of implicit places.

Consider an α-morphism $\varphi\colon N_1 \to N_2$ shown in Fig. 19, where transitions t_3, t_4 and t_5 in N_1 are local. N_2 cannot be obtained form N_1 (and vice versa) by applying the proposed rules. For instance, to apply the rule ρ_{A5}, the place p_2 has to be duplicated (the rule ρ_{R1}). However, even in this case, after fusing transitions t_1 and t_2 we will not be able to do pairwise simplification of transitions t_3, t_4 and t_5 since they do not have coincident presets and postsets.

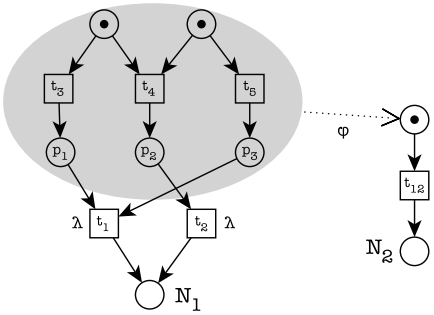


Fig. 19. An α-morphism that cannot be obtained by applying transformations

In this light, it is also rather interesting to study the completeness problem, for instance, to establish whether transformations allow us to generate all possible refinements of a given abstract EN system preserving its properties. Moreover, we also plan to characterize properties of irreducible EN systems.

References

1. van der Aalst, W.M.P.: Workflow verification: finding control-flow errors using petri-net-based techniques. In: van der Aalst, W., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 161–183. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45594-9_11
2. Bernardinello, L., Mangioni, E., Pomello, L.: Local State Refinement and Composition of Elementary Net Systems: An Approach Based on Morphisms. In: Koutny, M., van der Aalst, W.M.P., Yakovlev, A. (eds.) Transactions on Petri Nets and Other Models of Concurrency VIII. LNCS, vol. 8100, pp. 48–70. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40465-8_3
3. Bernardinello, L., De Cindio, F.: A survey of basic net models and modular net classes. In: Rozenberg, G. (ed.) Advances in Petri Nets 1992. LNCS, vol. 609, pp. 304–351. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55610-9_177
4. Bernardinello, L., Lomazova, I., Nesterov, R., Pomello, L.: Soundness-preserving composition of synchronously and asynchronously interacting workflow net components (2020). https://arxiv.org/pdf/2001.08064.pdf

5. Berthelot, G.: Checking properties of nets using transformations. In: Rozenberg, G. (ed.) Advances in Petri Nets 1985. LNCS, vol. 222, pp. 19–40. Springer, Heidelberg (1986). https://doi.org/10.1007/3-540-18086-9
6. Berthelot, G., Roucairol, G.: Reduction of Petri-nets. In: Mazurkiewicz, A. (ed.) Mathematical Foundations of Computer Science 1976. LNCS, vol. 45, pp. 202–209. Springer, Heidelberg (1976). https://doi.org/10.1007/978-1-4612-3086-1
7. Desel, J., Merceron, A.: Vicinity Respecting Homomorphisms for Abstracting System Requirements. In: Jensen, K., Donatelli, S., Koutny, M. (eds.) Transactions on Petri Nets and Other Models of Concurrency IV. LNCS, vol. 6550, pp. 1–20. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-18222-8_1
8. Ehrig, H., Hoffmann, K., Padberg, J.: Transformations of Petri nets. Electr. Notes Theor. Comput. Sci. **148**(1), 151–172 (2006)
9. Esparza, J., Silva, M.: Top-down synthesis of live and bounded free choice nets. In: Rozenberg, G. (ed.) ICATPN 1990. LNCS, vol. 524, pp. 118–139. Springer, Heidelberg (1991). https://doi.org/10.1007/BFb0019972
10. Genrich, H., Thiagarajan, P.: A theory of bipolar synchronisation schemes. Theor. Comput. Sci. **30**(3), 241–318 (1984)
11. Hack, M.: Analysis of Production Schemata by Petri Nets. TR-94. MIT Press, Boston (1972)
12. Lomazova, I.A.: Resource Equivalences in Petri Nets. In: van der Aalst, W., Best, E. (eds.) PETRI NETS 2017. LNCS, vol. 10258, pp. 19–34. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57861-3_3
13. Mikolajczak, B., Wang, Z.: Conceptual modeling of concurrent systems through stepwise abstraction and refinement using Petri net morphisms. In: Song, I., Liddle, S., Ling, T., Scheuermann, P. (eds.) Conceptual Modeling - ER 2003. LNCS, vol. 2813, pp. 433–445 (2003). https://doi.org/10.1007/b13244
14. Murata, T.: Petri nets: properties, analysis and applications. Proc. IEEE **77**(4), 541–580 (1989)
15. Murata, T., Suzuki, I.: A method for stepwise refinement and abstraction of petri nets. J. Comput. Syst. Sci. **27**(1), 51–76 (1983)
16. Nesterov, R., Lomazova, I.: Asynchronous interaction patterns for mining multi-agent system models from event logs. In: Lomazova, I., Kalenkova, A., Yavorsky, R. (eds.) Proceedings of the MACSPro Workshop 2019. CEUR Workshop Proceedings, vol. 2478, pp. 62–73. CEUR-WS.org (2019)
17. Nielsen, M., Winskel, G.: Petri nets and bisimulations. Theor. Comput. Sci. **153**, 211–244 (1996)
18. 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). https://doi.org/10.1007/978-3-540-40022-6_9
19. Rozenberg, G., Engelfriet, J.: Elementary net systems. In: Reisig, W., Rozenberg, G. (eds.) ACPN 1996. LNCS, vol. 1491, pp. 12–121. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-65306-6_14
20. Schnoebelen, P., Sidorova, N.: Bisimulation and the reduction of Petri nets. In: Nielsen, M., Simpson, D. (eds.) Application and Theory of Petri Nets 2000. LNCS, vol. 1825, pp. 409–423. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44988-4
21. Valette, R.: Analysis of Petri nets by stepwise refinements. J. Comput. Syst. Sci. **18**(1), 35–46 (1979)
22. Winskel, G.: Petri nets, algebras, morphisms, and compositionality. Inf. Comput. **72**(3), 197–238 (1987)