Property-Preserving Transformations of Elementary Net Systems Based on Morphisms*

Luca Bernardinello[2], Irina Lomazova[1], Roman Nesterov[1,2], Lucia Pomello[2]

[1] National Research University Higher School of Economics,
20 Myasnitskaya Ulitsa, 101000 Moscow, Russia
[2] Dipartimento di Informatica, Sistemistica e Comunicazione,
Università degli Studi di Milano-Bicocca,
Viale Sarca 336 - Edificio U14, I-20126 Milano, Italia

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. We also show 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 is a convenient formalism for modeling concurrent systems as well as for proving their important behavioral properties[1]. 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

[1] This work is supported by MIUR and the Basic Research Program at the National Research University Higher School of Economics.

local structural reductions and extensions, s.t. liveness, boundedness (safeness), covering by place invariants, home state and proper termination can be preserved by corresponding transformations.

A class of free choice Petri nets [11] is also widely adopted to model 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 graphs of 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, 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 and minimizing unlabeled ones. These transformations are such that an initial net and a transformed net are related by an α-morphism, and their behavioral properties including reachable markings and, especially, deadlocks are preserved. Interestingly enough, it is also shown that simple Petri net transformations described earlier also yield corresponding α-morphisms.

In addition, we demonstrate 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 formal models of abstract interaction patterns [16] that give ready-to-use solutions to organize correct interactions of components in complex parallel systems.

This paper is organized as follows. The next section gives the basic definitions. In Section 3, we define local abstraction and refinement transformations for EN

systems, s.t. they induce α-morphisms. Section 4 discusses application of defined transformations to the problem of constructing correct systems with interacting workflow net components, whereas Section 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 are depicted by boxes, and a flow relation is shown by arcs.

Let $N = (P, T, F)$ be a net. The *preset* of $x \in P \cup T$ is the set ${}^\bullet x = \{y \in P \cup T \,|\, (y, x) \in F\}$. The *postset* of $x \in P \cup T$ is the set $x^\bullet = \{y \in P \cup T \,|\, (x, y) \in F\}$. 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, s.t. $\forall t \in T\colon |{}^\bullet t| \geq 1$ 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 ${}^\bigcirc N(Y) = \{y \in Y \,|\, {}^\bullet y = \varnothing\}$ is the *input* border, and the set $N(Y)^\bigcirc = \{y \in Y \,|\, y^\bullet = \varnothing\}$ is the *output* border of a subnet $N(Y)$.

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).

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 a marking m is denoted by $[m\rangle$. It can be easily checked that reachable markings in an SMD-EN system

are free from contacts. A reachable marking m is a *deadlock* if it does not enable any transition.

A deadlock m in an SMD-EN system $N = (P, T, F, m_0)$ can be characterized from a structural point of view. Let $t \in T$, s.t. ${}^\bullet t \cap m \neq \varnothing$, and $p \in P$, s.t. $p \in {}^\bullet t$ and $p \notin m$. There is a sequential component $N(C \cup {}^\bullet C^\bullet)$ of N containing p, i.e., $p \in C$. Let $p' = m \cap C$. Then either $(p')^\bullet = \varnothing$ and we are done, or we take a transition $t' \in (p')^\bullet$ and repeat the same step we have done for t. This procedure will finish either in a place with an empty postset or in a transition it started with. Intuitively, a deadlock is a poor synchronization of sequential components in N, where they are "waiting" for each other.

For instance, the EN system shown in Fig. 1 has two deadlocks $\{p_1, p_4\}$ and $\{p_2, p_3\}$, where the left and the right sequential components are waiting for each other since conflicts have been resolved independently.

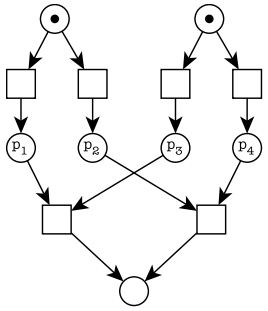


Fig. 1. Deadlock as poorly synchronized sequential components

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. $\forall t_1 \in T_1$: *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. $\forall t_1 \in T_1$: *if* $\varphi(t_1) \in P_2$, *then* $\varphi({}^\bullet t_1{}^\bullet) = \{\varphi(t_1)\}$.
5. $\forall p_2 \in P_2$:
   (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) $\forall p_1 \in P_1 \cap \varphi^{-1}(p_2)$: *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. Requirements imposed by Definition 1 lead to the following observation. If a subnet in N_1 refines a place in N_2, then this subnet should behave exactly as an abstract place does. More precisely, (a) no tokens are left in a subnet refining a place after firing a transition in a postset of an output border; (b) no transitions are enabled in the preset of an input border of a subnet refining a place whenever a token is inside a subnet. Main properties preserved and reflected by α-morphisms have been studied in [2]. We will mention them further in the paper where necessary.

Figure 2, borrowed from [4], shows an example of α-morphism, where N_1 is a refinement of an abstract net 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 in N_1.

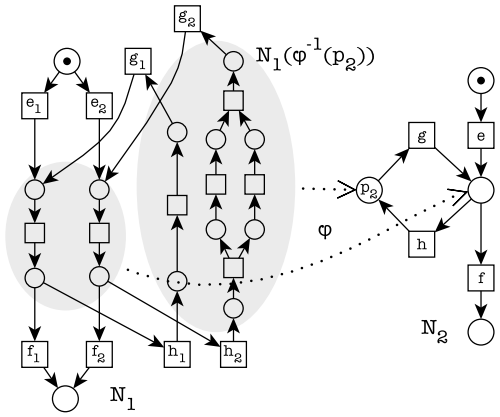


Fig. 2. An α-morphism $\varphi\colon N_1 \to N_2$

3 Structural Transformations of SMD-EN Systems

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 asyn-

chronously 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 τ 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 that define applicability of a rule to an EN system.
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.

Constraints c_L and c_R are necessary to construct an α-morphism between the initial and the transformed EN system.

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 $X_L \subseteq P \cup T$, such that 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 neighborhood ${}^\bullet X_L^\bullet$.
3. Make necessary changes, i.e., relabel transitions and add tokens to places, in an inserted subnet according to c_R.

Correspondingly, the effect of applying ρ to a subnet $N(X_L)$ in N isomorphic to the left part of ρ is denoted by $\rho(N, X_L) = (P', T', F', m'_0)$ with a new transition labeling function $h' \colon T' \to \Lambda$.

3.1 Abstraction Rules

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

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).

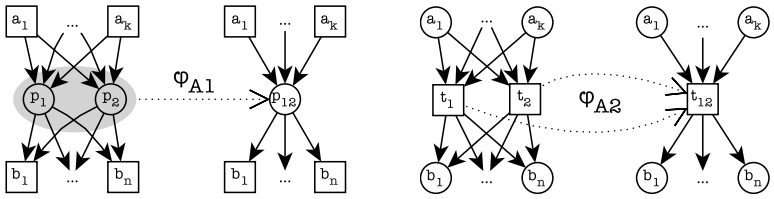
- *transformation:* fusion of p_1 and p_2 into a single place p_{12}, s.t. ${}^\bullet p_{12} = {}^\bullet p_1 = {}^\bullet p_2$, $p_{12}{}^\bullet = p_1{}^\bullet = p_2{}^\bullet$ and $p_{12} \in m_0' \Leftrightarrow (p_1 \in m_0$ and $p_2 \in m_0)$.
- *α-morphism* $\varphi_{A1} \colon N \to N'$, where $N' = \rho_{A1}(N, \{p_1, p_2\})$, maps places p_1 and p_2 in N to a 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 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 $h(t_1) = h(t_2)$) as demonstrated in Fig. 3(b)).
- *transformation:* fusion of t_1 and t_2 into a single transitions t_{12}, s.t. ${}^\bullet t_{12} = {}^\bullet t_1 = {}^\bullet t_2$, $t_{12}{}^\bullet = t_1{}^\bullet = t_2{}^\bullet$ 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 a 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) has also been mentioned 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}, s.t. ${}^\bullet p_{12} = {}^\bullet p_1$, $p_{12}{}^\bullet = p_2{}^\bullet$ 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 a single 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}. Applicability constraints of ρ_{A3} are aimed to avoid generating isolated places and self-loops in $\rho_{A3}(N, \{p_1, t, p_2\})$. A similar transition transformation called "pre-fusion" has been discussed in [5], where it has been expressed as fusion of two transitions connected by a place.

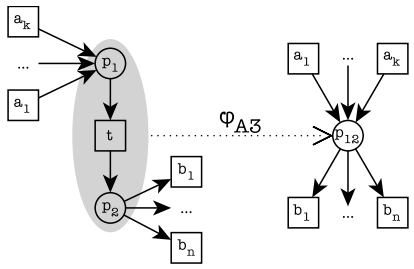


Fig. 4. Abstraction rule A3: local transition elimination

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 left parts of rules together with 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 $p_2 \in m_0)$ as illustrated in 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 a single 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.

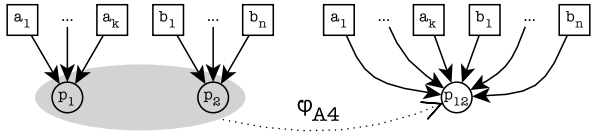


Fig. 5. Abstraction rule A4: Postset-empty place simplification

In the following 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 described in [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. The net system shown in Fig. 1 has two deadlocks $\{p_1, p_4\}$ and $\{p_2, p_3\}$. These deadlocks are caused by the fact that conflicts are resolved independently by two sequential components. Suppose that the two transitions t_1 and t_2 in the postsets of $p_1, \ldots, p_4$ have the same label. Then, according to the definition of α-morphisms, it is possible to fuse p_1 with p_2, p_3 with p_4 and t_1 with t_2 correspondingly. The image t' of t_1 and t_2 will have two places in its preset, and there will be the reachable marking $\{p_{12}, p_{34}\}$ in a transformed net system enabling t'. However, there is an inverse image of the marking $\{p_{12}, p_{34}\}$, e.g., the deadlock $\{p_1, p_4\}$ that does not enable an inverse image of t'.

Correspondingly, we impose constraints on places in the presets of two transitions to be fused, s.t. if there is a deadlock containing places in the presets of both transitions, then it should not be possible to fuse these transitions.

A5: Preset-disjoint transition simplification

- *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. $\forall a \in {}^\bullet t_1 \, \forall b \in {}^\bullet t_2 \, \exists C \subseteq P \colon a, b \in C$, $N(C \cup {}^\bullet C^\bullet)$ 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. As for the initial marking m'_0 in $\rho_{A5}(N, \{t_1, t_2\})$, we have $\forall (a, b) \in {}^\bullet t_{12} \colon (a, b) \in m'_0 \Leftrightarrow ((a \in m_0$ and $b \notin m_0)$ or $(a \notin m_0$ and $b \in m_0))$.
- *α-morphism* $\varphi_{A5} \colon N \to N'$, where $N' = \rho_{A5}(N, \{t_1, t_2\})$, maps t_1 and t_2 to t_{12} in N' as well as pairs of places in ${}^\bullet t_1$ and ${}^\bullet t_2$ corresponding to each other by g, and for other nodes in N, φ_{A5} is the identity mapping.

In Fig. 6, we provide left and right parts of the abstraction rule ρ_{A5}.

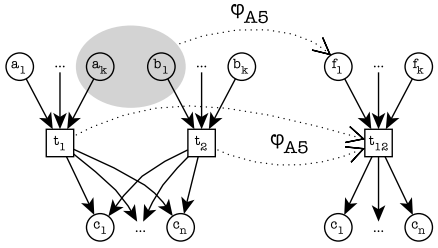


Fig. 6. Abstraction rule A5: Preset-disjoint transition simplification

The third requirement of ρ_{A5} makes sure that any place in ${}^\bullet t_1$ is in conflict with any 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.

Application of this abstraction rule involves pairwise place fusion in ${}^\bullet t_1$ and ${}^\bullet t_2$. According to the third 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 a more detailed example of applying the abstraction rule ρ_{A5}. Two transitions t_1 and t_2 in a net system N_1 shown in Fig. 7 are candidates to be fused, since they have the same label λ and share the same postset. We need to check whether places in ${}^\bullet t_1$ and ${}^\bullet t_2$ are connected by sequential components (for short, SC):

1. p_1 and p_3 are in the SC generated by $C = \{s_4, s_1, p_1, p_3, p_5\}$ and ${}^\bullet C^\bullet$.
2. p_2 and p_3 are in the SC generated by $C = \{s_4, s_2, p_2, p_3, p_5\}$ and ${}^\bullet C^\bullet$.
3. p_2 and p_4 are in the SC generated by $C = \{s_5, s_3, p_2, p_4, p_5\}$ and ${}^\bullet C^\bullet$.
4. p_1 and p_4 together do not belong to any SC.

Indeed, there is a deadlock $\{p_1, s_2, p_4\}$, s.t. it contains places both from ${}^\bullet t_1$ and from ${}^\bullet t_2$. Thus, t_1 and t_2 cannot be fused without losing the deadlock.

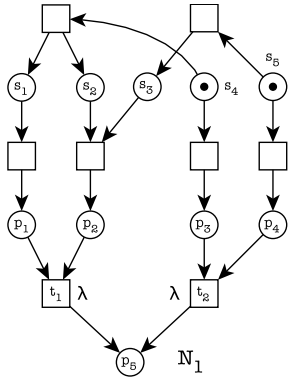


Fig. 7. A net system to check applicability constraints of ρ_{A5}

It is worth mentioning that application of the rule ρ_{A5} can also be straightforwardly extended to the case when transitions t_1 and t_2 have shared places in their presets.

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

By construction, application of an abstraction rule induce an α-morphism from an initial SMD-EN system towards a 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 $\varphi_{\rho_2} \circ \varphi_{\rho_1} \colon N \to \rho_2(\rho_1(N, X_L), X'_L)$.*

The important property is whether the order of applying abstraction rules matter 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 $N(X_L^1)$ in N, ρ_2 is applicable to a subnet $N(X_L^2)$ in N and $X_L^1 \neq X_L^2$. 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 $\rho_2(\rho_1(N, X_L^1), X_L^2) = \rho_1(\rho_2(N, X_L^2), X_L^1)$.*

According to the structural characterization of a deadlock in an SMD net system and to the structural requirements of abstraction rules, we 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. In proving this statement, we also rely on the fact that α-morphisms *preserve* reachable markings and transition firings [2], 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 $\varphi_\rho(t) = \varphi_\rho(m)$, i.e., a transition t and ${}^\bullet t^\bullet$ is mapped to a single 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 it is a contradiction.

3.2 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.

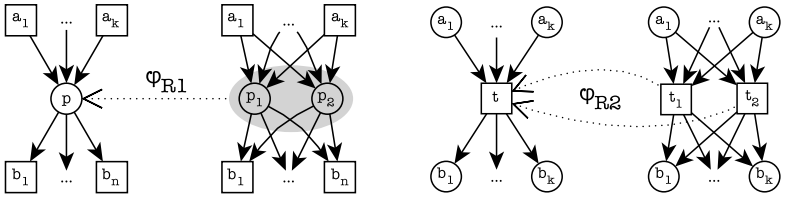
For what follows, let $N = (P, T, F, m_0)$ be an SMD-EN system with a transition labeling function $h : T \to \Lambda \cup \{\tau\}$. Recall also that the effect of applying a transformation rule ρ to N is denoted by $\rho(N, X_L) = (P', T', F', m_0')$ with a new transition labeling function $h'\colon T' \to \Lambda \cup \{\tau\}$, where $X_L \subseteq P \cup T$ and $N(X_L)$ is a corresponding 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, s.t. ${}^\bullet p_1 = {}^\bullet p_2 = {}^\bullet p$, $p_1{}^\bullet = p_2{}^\bullet = p^\bullet$ and ($p_1 \in m'_0$ and $p_2 \in m'_0$) $\Leftrightarrow p \in m_0$ (see Fig. 8(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 a single 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, s.t. ${}^\bullet t_1 = {}^\bullet t_2 = {}^\bullet t$ and $t_1{}^\bullet = t_2{}^\bullet = t^\bullet$ (see Fig. 8(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 a single 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. 8. Place and transition duplication

R3: Local transition introduction

- *applicability constraints:* a place p in N.
- *transformation:* substitution of p with a transition t (see Fig. 9), s.t.:
  1. ${}^\bullet t = \{p_1\}$ and $t^\bullet = \{p_2\}$;
  2. $p_1{}^\bullet = {}^\bullet p_2 = \{t\}$;
  3. ${}^\bullet p_1 = {}^\bullet p$ and $p_2{}^\bullet = p^\bullet$;
  4. $p_1 \in m'_0 \Leftrightarrow p \in m_0$.
- *α-morphism* $\varphi_{R3}\colon N' \to N$, where $N' = \rho_3(N, \{p\})$, maps t, p_1 and p_2 in N' to a single place p in N. For other nodes in N', φ_{R3} is the identity mapping between N' and N.

The refinement rules ρ_{R1}, ρ_{R2} and ρ_{R3}, similar to their abstraction counterparts ρ_{A1}, ρ_{A2} and ρ_{A3}, can be easily generalized.

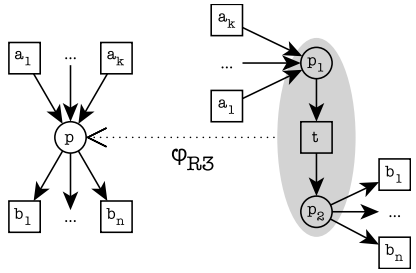


Fig. 9. Refinement rule R3: local transition introduction

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. 10), s.t.:
  1. ${}^\bullet p_1 \subset {}^\bullet p$ and ${}^\bullet p_2 \subset {}^\bullet p$;
  2. ${}^\bullet p_1 \cap {}^\bullet p_2 = \varnothing$ and ${}^\bullet p_1 \cup {}^\bullet p_2 = {}^\bullet p$;
  3. $|p_1{}^\bullet| = |p_2{}^\bullet| = |p^\bullet|$, and there is a bijection $f_i \colon p_i{}^\bullet \to p^\bullet$ with $i = 1, 2$, s.t. if $f_i(x) = b$ for $x \in p_i{}^\bullet$ and $b \in p^\bullet$, then $h'(x) = h(b)$;
  4. ${}^\bullet(p_i{}^\bullet) \setminus \{p_i\} = {}^\bullet(p^\bullet) \setminus \{p\}$ with $i = 1, 2$;
  5. 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 a single place p in N. For other nodes in N', φ_{R4} is the identity mapping between N' and N.

While splitting a place p in N, we also split its neighborhood between p_1 and p_2 in $\rho_{R4}(N, \{p\})$. According to constraints 1 and 2, the preset of p is divided into two disjoint, proper and non-empty subsets. According to constraints 3 and 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 constraint 4, if there are other input places to the postset of p in N, then they are also input places to the postsets of p_1 and p_2 in $\rho_{R4}(N, \{p\})$. These requirements on splitting the neighborhood of p in N are based on the requirements 5b and 5c of Definition 1 (α-morphisms).

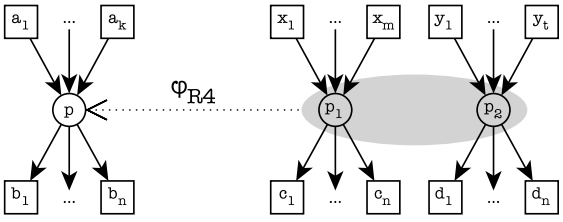


Fig. 10. Refinement rule R4: place split

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.

By construction, application of the refinement rules induces an α-morphisms 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., any deadlock in a refined EN system is an image of 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. Follows from the structural characterization of a deadlock in an SMD-EN system and from the fact that application of the refinement rules, that results in splitting places (thus generating new images of reachable markings in an initial model N), fully preserves their neighborhoods.

4 Use of Transformations for Workflow Net Composition

In this section, we demonstrate application of transformations defined in the previous section to a correct composition of interacting workflow net components. Workflow nets have both initial and final markings. We follow an approach to a correct composition of *generalized workflow nets* (GWF-nets) described in [4], and 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 formal 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. $\forall t \in T\, \exists m \in [m_0\rangle \colon m[t\rangle$.

The composition of two GWF-nets N_1 and N_2 is a GWF-net denoted by $N_1 \circledast N_2$, s.t. it is fully defined according to transition labels: (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 (sending and receiving).

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

Proposition 6. *Let R_1, R_2 and A_1, A_2 be four sound 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 $R_1 \circledast R_2$ sound.*

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

4.1 Abstraction of Interacting Workflow Net Components

Here we show application of abstraction rules to build abstract representations of interacting GWF-net components. Fro example, we construct the α-morphism shown in Fig. 2 step by step. Transitions $e_1, e_2, f_1, f_2, g_1, g_2, h_1, h_2$ in N_1 are considered to be 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.

Firstly, local transitions $y_1, \ldots, y_5$ can be eliminated using the rule ρ_{A3} for five times. After collapsing these local transitions, we simplify places p_1 and p_2 (the rule ρ_{A1}) that are generated from eliminating local transitions y_4 and y_5 correspondingly. Figure 11 gives a concise illustration of these transformations.

Now local transitions y_7 and y_8 are also eliminated using the rule ρ_{A3} twice (see the transformation 3 in Fig. 12). Unfortunately, the fourth transformation shown in Fig. 12 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 bee seen as an example of a direct application of Definition 1 by constructing appropriate fusions. We plan to investigate possible local transformations behind this in the future.

Finally, we simplify transitions e_1 and e_2 (abstraction rule ρ_{A2}) and obtain the target abstract EN system previously shown 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.

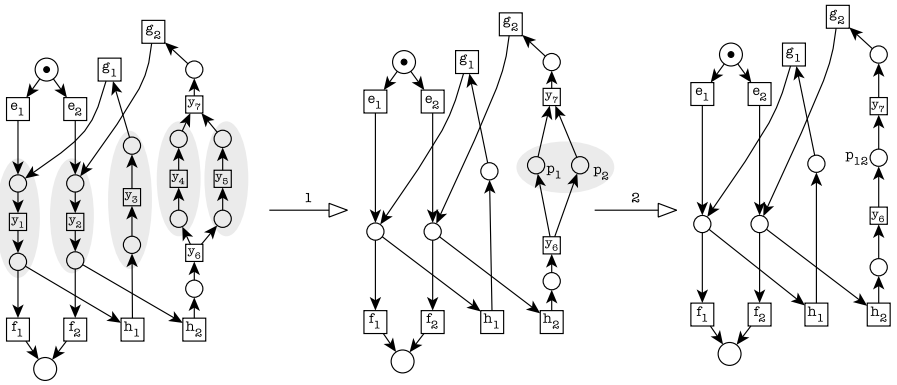


Fig. 11. Abstracting a GWF-net: steps 1 and 2

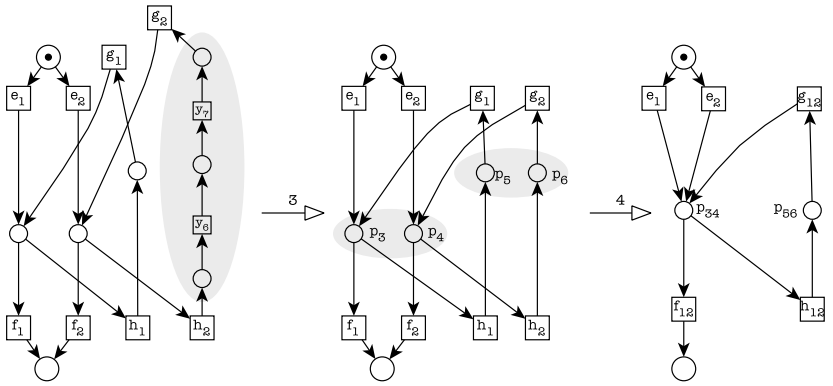


Fig. 12. Abstracting a GWF-net: steps 3 and 4

4.2 Refinement of Interaction Patterns

In this section, we apply the refinement rules defined in the previous section and solve the inverse problem: the construction of a detailed system model that preserves properties of an initial abstract model.

As described above, a composition of abstract representations of interacting GWF-net components models an interaction pattern GWF-net components communicate according to. In [16], typical interaction patterns for asynchronously communicating GWF-net components have been discussed. Abstract interaction patterns provide generic solutions that can be used to model and verify component interactions in large-scale distributed systems. Accordingly, a formal model of an abstract interaction pattern is a composition of GWF-nets.

Consider a pattern shown in Fig. 13 (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 simple rules.

Figure 14 gives a concise illustration for building a possible refinement of the “Send-Receive“ interaction pattern. We describe applied refinement rules below.

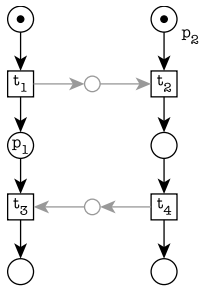


Fig. 13. "Send-Receive" asynchronous interaction pattern

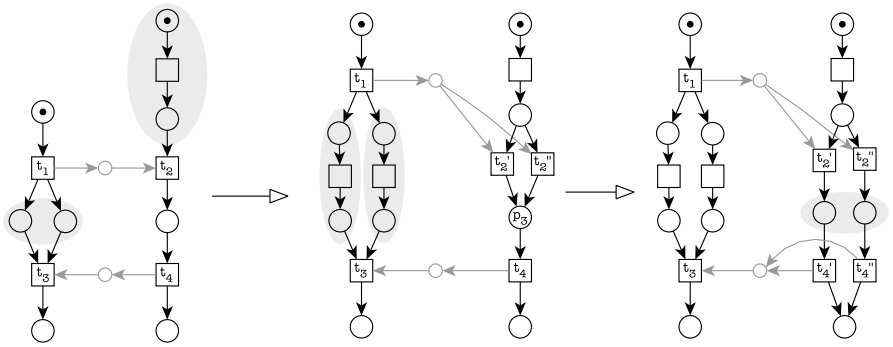


Fig. 14. Refinement of the interaction pattern from Fig. 13

Firstly, we duplicate the place p_1 (rule ρ_{R1}) and introduce a local transition instead of the place p_2 (rule ρ_{R3}). Secondly, we also introduce local transitions instead of copies of the place p_1, and we duplicate the transition t_2 (rule ρ_{R2}). The last transformation shown in Fig. 14 is the split of the place p_3 according to the refinement rule ρ_{R4}. Refinement process may be continued until the target detail level is achieved.

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. Step-wise application of these transformations 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 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. In this light, it is also rather interesting to study the completeness problem, for instance, to establish whether transformations allows 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: Models, Techniques, and Empirical Studies. LNCS, vol. 1806, pp. 161–183. Springer, Heidelberg (2000)
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., Yakovlev, A. (eds.) ToPNoC VIII. LNCS, vol. 8100, pp. 48–70. Springer, Heidelberg (2013)
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)
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)
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)
7. Desel, J., Merceron, A.: Vicinity respecting homomorphisms for abstracting system requirements. In: Jensen, K., Donatelli, S., Koutny, M. (eds.) ToPNoC IV. LNCS, vol. 6550, pp. 1–20. Springer, Heidelberg (2010)
8. Ehrig, H., Hoffmann, K., Padberg, J.: Transformations of Petri nets. Electronic Notes in Theoretical Computer Science 148(1), 151–172 (2006)
9. Esparza, J., Silva, M.: Top-down synthesis of live and bounded free choice nets. In: Rozenberg, G. (ed.) Advances in Petri Nets 1991. LNCS, vol. 524, pp. 118–139. Springer, Heidelberg (1991)
10. Genrich, H., Thiagarajan, P.: A theory of bipolar synchronisation schemes. Theoretical Computer Science 30(3), 241–318 (1984)
11. Hack, M.: Analysis of production schemata by Petri nets. TR-94. MIT, Boston (1972)

12. Lomazova, I.A.: Resource equivalences in Petri nets. In: van der Aalst, W., Best, E. (eds.) Application and Theory of Petri Nets and Concurrency. LNCS, vol. 10258, pp. 19–34. Springer, Heidelberg (2017)
13. Mikolajczak, B., Z., W.: 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)
14. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (1989)
15. Murata, T., Suzuki, I.: A method for stepwise refinement and abstraction of petri nets. Journal of Computer and System Sciences 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. Theoretical Computer Science 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: Advances in Petri Nets. LNCS, vol. 2472, pp. 161–196. Springer, Heidelberg (2003)
19. Rozenberg, G., Engelfriet, J.: Elementary net systems. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models: Advances in Petri Nets. LNCS, vol. 1491, pp. 12–121. Springer, Heidelberg (1998)
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)
21. Valette, R.: Analysis of Petri nets by stepwise refinements. Journal of Computer and System Sciences 18(1), 35–46 (1979)
22. Winskel, G.: Petri nets, algebras, morphisms, and compositionality. Information and Computation 72(3), 197–238 (1987)