Channel Properties of Asynchronously Composed Petri Nets*

Serge Haddad[1], Rolf Hennicker[2], and Mikael H. Møller[3]

[1] LSV, ENS Cachan & CNRS & INRIA, France
[2] Ludwig-Maximilians-Universität München, Germany
[3] Aalborg University, Denmark

Abstract. We consider asynchronously composed I/O-Petri nets (AIOPNs) with built-in communication channels. They are equipped with a compositional semantics in terms of asynchronous I/O-transition systems (AIOTSs) admitting infinite state spaces. We study various channel properties that deal with the production and consumption of messages exchanged via the communication channels and establish useful relationships between them. In order to support incremental design we show that the channel properties considered in this work are preserved by asynchronous composition, i.e. they are compositional. As a crucial result we prove that the channel properties are decidable for AIOPNs.

1 Introduction

(A)synchronous composition. The design of hardware and software systems is often component-based which has well-known advantages: management of complexity, reusability, separation of concerns, collaborative design, etc. One critical feature of such systems is the protocol supporting the communication between components and, in particular, the way they synchronise. Synchronous composition ensures that both parts are aware that communication has taken place and then simplifies the validation of the system. However in a large scale distributed environment synchronous composition may lead to redhibitory inefficiency during execution and thus asynchronous composition should be adopted. The FIFO requirement of communication channels is often not appropriate in this context. This is illustrated by the concept of a software bus where applications push and pop messages in mailboxes. Also on the modelling level FIFO ordering is often not assumed, like for the composition of UML state machines which relies on event pools without specific requirements.

Compositions of Petri nets. In the context of Petri nets, composition has been studied both from theoretical and practical points of view. The process algebra approach has been investigated by several works leading to the Petri net algebra [4]. Such an approach is closely related to synchronous composition. In [22] and [23] asynchronous composition of nets are performed via a set of places or, more generally, via a subnet modelling some medium. Then structural

* This work has been partially sponsored by the EU project ASCENS, 257414.

restrictions on the subnets are proposed in order to preserve global properties like liveness or deadlock-freeness. In [21] a general composition operator is proposed and its associativity is established. A closely related concept to composition is the one of open Petri nets which has been used in different contexts like the analysis of web services [25]. Numerous compositional approaches have been proposed for the modelling of complex applications but most of them are based on high-level Petri nets; see [11] for a detailed survey.

Channel properties. With the development of component-based applications, one is interested in verifying behavioural properties of the communication and, in the asynchronous case, in verifying the properties related to communication channels. Channel properties naturally occur when reasonning about distributed mechanisms, algorithms and applications (e.g. management of sockets in UNIX, maintaining unicity of a token in a ring based algorithm, recovery points with empty channels for fault management, guarantee of email reading, etc.).

Our contributions. In this work we are interested in general channel properties and not in specific system properties related to particular applications. The FIFO requirement for channels potentially can decrease the performance of large scale distributed systems. Thus we restrict ourselves to *unordered* channels which can be naturally modelled by places of Petri nets. We propose asynchronously composed Petri nets (AIOPNs) by (1) explicitely representing channels for internal communication inside the net and (2) defining communication capabilities to the outside in terms of (open) input and output labels with appropriate transitions. Then we define an asynchronous composition operator which introduces new channels for the communication between the composed nets. AIOPNs are equipped with a semantics in terms of asynchronously composed I/O- transition systems (AIOTS). We show that this semantics is fully compositional, i.e. it commutes with asynchronous composition.

In our study two kinds of channel properties are considered which are related to consumption requirements and to the termination of communication. Consumption properties deal with requirements that messages sent to a communication channel should also be consumed. They can be classified w.r.t. two criteria. The first criterion is the nature of the requirement: consuming messages, decreasing the number of messages on a channel, and emptying channels. The second criterion expresses the way the requirement is achieved: possibly immediately, possibly after some delay, or necessarily in each weakly fair run. Communication termination deals with (immediate or delayed) closing of communication channels when the receiver is not ready to consume any more. We establish useful relations between the channel properties and prove that all channel properties considered here are compositional, i.e. preserved by asynchronous composition, which is an important prerequisite for incremental design.

From a verification point of view, we study the decidability of properties in the framework of AIOPN. Thanks to several complementary works on decidability for Petri net problems, we show that all channel properties considered in this work are decidable, though with a high computational complexity.

Related work. To the best of our knowledge, no work has considered channels explicitely defined for communication inside composite components. However there have been several works where channels are associated with asynchronous composition. They can be roughly classified depending whether their main feature is an algorithmic or a semantic one.

From an algorithmic point of view, in the seminal work of [5], the authors discus several properties like *channel boundedness* and *specified receptions* and propose methods to analyse them. In [7], a two-component based system is studied using a particular (decidable) channel property, the *half-duplex property*: at any time at most one channel is not empty.

From a semantic point view, in [13] "connection-safe" component assemblies have been studied incorporating both synchronous and asynchronous communication. More recently in [2] *synchronisability*, a property of asynchronous systems, is introduced such that when it holds the system can be safely abstracted by a synchronous one. In the framework of Petri nets, E. Kindler has defined Petri net components where the interface is composed by places and composition consists in merging places with same identities [16]. He proposed a partial order semantics for such components and proved that the semantic is fully compositional. Furthermore, for a restricted linear temporal logic he established that properties of this logic are preserved by composition; however such a logic cannot express some of the channel properties we introduce here due to their branching kind.

The Petri net based formalism of open nets is the closest formalism to ours. Several works [17],[25], and [24] address both the semantic point of view and the algorithmic one but only when the nets are assumed to be bounded which is not required here. We postpone to Section 2.2 a more detailed comparison with our formalism.

Organisation. In Section 2, we introduce AIOPNs and their asynchronous composition. Then, we provide a compositional semantic for AIOPNs in Section 3 in terms of AIOTSs. In Section 4, we define the channel properties and study their relationships and their preservation under asynchronous composition. In Section 5, we establish that all channel properties are decidable for AIOPNs. Finally, in Section 6, we conclude and give some perspectives for future work.

2 Asynchronous I/O-Petri Nets

2.1 Basic Notions

We recall some basic notions of labelled Petri nets and define their transition system semantics. A *labelled Petri net* is a tuple $\mathcal{N} = (P, T, \Sigma, W^-, W^+, \lambda, m^0)$, such that

- P is a finite set of places,
- T is a finite set of transitions with $P \cap T = \emptyset$,
- Σ is a finite alphabet,

– W^- (resp. W^+) is a matrix indexed by $P \times T$ with values in $\mathbb{N}$;
  it is called the *backward* (resp. *forward*) *incidence matrix*,
– $\lambda : T \to \Sigma$ is a transition labelling function, and
– m^0 is a vector indexed by P and called the initial marking.

The labelling function λ is extended as usual to sequences of transitions. The input (output resp.) vector $W^-(t)$ ($W^+(t)$ resp.) of a transition t is the column vector of matrix W^- (W^+ resp.) with index t. Given two vectors $\overrightarrow{v}$ and $\overrightarrow{v}'$, one writes $\overrightarrow{v} \geq \overrightarrow{v}'$ if $\overrightarrow{v}$ is componentwise greater or equal than $\overrightarrow{v}'$. *A marking* is a vector indexed by P. A transition $t \in T$ is *firable* from a marking m, denoted by $m \xrightarrow{t}$, if $m \geq W^-(t)$. The firing of t from m leads to the marking m', denoted by $m \xrightarrow{t} m'$, and defined by $m' = m - W^-(t) + W^+(t)$. If $\lambda(t) = a$ we write $m \xrightarrow{a} m'$. The firing of a transition is extended as usual to firing sequences $m \xrightarrow{\sigma} m'$ with $\sigma \in T^*$. A marking m is reachable if there exists a firing sequence $\sigma \in T^*$ such that $m^0 \xrightarrow{\sigma} m$.

Our approach is based on a state transition system semantics for Petri nets. A *labelled transition system* (LTS) is a tuple $\mathcal{S} = (\Sigma, Q, q^0, \longrightarrow)$, such that

– Σ is a finite set of labels,
– Q is a (possibly infinite) set of states,
– $q^0 \in Q$ is the initial state, and
– $\longrightarrow \subseteq Q \times \Sigma \times Q$ is a labelled transition relation.

We will write $q \xrightarrow{a} q'$ for $(q, a, q') \in \longrightarrow$, and we write $q \xrightarrow{a}$ if there exists $q' \in Q$ such that $q \xrightarrow{a} q'$. Let $q_1 \in Q$. A *trace* of $\mathcal{S}$ starting in q_1 is a finite or infinite sequence $\rho = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \xrightarrow{a_3} \cdots$. For $a \in \Sigma$ we write $a \in \rho$, if there exists a_i in the sequence ρ such that $a_i = a$, and $\sharp_\rho(a)$ denotes the (possibly infinite) number of occurrences of a in ρ. For $q \in Q$ we write $q \in \rho$, if there exists q_i in the sequence ρ such that $q_i = q$. For $\sigma = a_1 a_2 \cdots a_n \in \Sigma^*$ and $q, q' \in Q$ we write $q \xrightarrow{\sigma} q'$ if there exists a (finite) trace $q \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q'$. Often we need to reason about the successor states reachable from a given state $q \in Q$ with a subset of labels $\bar{\Sigma} \subseteq \Sigma$. We define $\mathrm{Post}(q, \bar{\Sigma}) = \{q' \in Q \mid \exists a \in \bar{\Sigma} \,.\, q \xrightarrow{a} q'\}$ and we write $\mathrm{Post}(q)$ for $\mathrm{Post}(q, \Sigma)$. Further we define $\mathrm{Post}^*(q, \bar{\Sigma}) = \{q' \in Q \mid \exists \sigma \in \bar{\Sigma}^* \,.\, q \xrightarrow{\sigma} q'\}$ and we write $\mathrm{Post}^*(q)$ for $\mathrm{Post}^*(q, \Sigma)$.

The semantics of a labelled Petri net $\mathcal{N} = (P, T, \Sigma, W^-, W^+, \lambda, m^0)$ is given by its *associated* labelled transition system $\mathsf{lts}(\mathcal{N}) = (\Sigma, Q, q^0, \longrightarrow)$ which represents the reachability graph of the net and is defined by

– $Q \subseteq \mathbb{N}^P$ is the set of reachable markings of $\mathcal{N}$,
– $\longrightarrow = \{(m, a, m') \mid a \in \Sigma \text{ and } m \xrightarrow{a} m'\}$, and
– $q^0 = m^0$.

2.2 Asynchronous I/O-Petri Nets and Their Composition

In this paper we consider systems which may be open for communication with other systems and may be composed to larger systems. Both the behaviour of

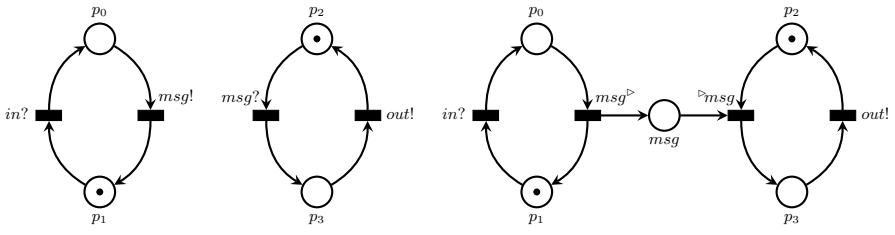
primitive components *and* of larger systems obtained by composition can be described by asynchronous I/O-Petri nets introduced in the following. We assume that communication is asynchronous and takes place via unbounded and unordered channels such that for each message type to be exchanged within a system there is exactly one communication channel. The open actions are modelled by distinguished input and output labels while communication inside the system via the channels is modelled by communication labels. Given a finite set C of channels, an *I/O-alphabet over* C is the disjoint union $\Sigma = \mathsf{in} \uplus \mathsf{out} \uplus \mathsf{com}$ of pairwise disjoint sets in of input labels, out of output labels and com of communication labels, such that $\Sigma \cap C = \emptyset$, $\mathsf{com} = \{ {}^{\triangleright}a, a^{\triangleright} \mid a \in C\}$ and in and out do not contain labels of the form ${}^{\triangleright}x$ or $x^{\triangleright}$. For each channel $a \in C$, the communication label ${}^{\triangleright}a$ represents consumption of a message from the channel a and $a^{\triangleright}$ represents putting a message on a . Each channel is modelled as a place and the transitions for communication actions are modelled by putting or removing tokens from the channel places. Three examples of AIOPNs are shown in Fig. 1. The nets $\mathcal{N}_1$ and $\mathcal{N}_2$ model primitive components (without channels) which repeatedly input and output messages. The net $\mathcal{N}_3$ in Fig. 1c models a simple producer/consumer system with one channel msg obtained by composition of the two primitive components; see below. Here and in the following drawings input labels are indicated by "?" and output labels by "!".

Definition 1 (Asynchronous I/O-Petri net). *An* asynchronous I/O-Petri net *(AIOPN) is a tuple* $\mathcal{N} = (C, P, T, \Sigma, W^-, W^+, \lambda, m^0)$*, such that*

- $(P, T, \Sigma, W^-, W^+, \lambda, m^0)$ *is a labelled Petri net,*
- C *is a finite set of channels,*
- $C \subseteq P$*, i.e. each channel is a place,*
- $\Sigma = \mathsf{in} \uplus \mathsf{out} \uplus \mathsf{com}$ *is an I/O-alphabet over* C*,*
- *for all* $a \in C$ *and* $t \in T$*,*

$$W^-(a,t) = \begin{cases} 1 & \text{if } \lambda(t) = {}^{\triangleright}a, \\ 0 & \text{otherwise} \end{cases} \qquad W^+(a,t) = \begin{cases} 1 & \text{if } \lambda(t) = a^{\triangleright}, \\ 0 & \text{otherwise} \end{cases}$$

- *for all* $a \in C$*,* $m^0(a) = 0$*.* ♢



(a) $\mathcal{N}_1$. (b) $\mathcal{N}_2$. (c) $\mathcal{N}_3$.

Fig. 1: Asynchronous I/O-Petri nets.

Two I/O-alphabets are composable if there are no name conflicts between labels and channels and, following [1], if shared labels are either input labels of one alphabet and output labels of the other or conversely. For the composition each shared label a gives rise to a new communication channel, also called a, and hence to new communication labels $a^{\triangleright}$ for putting and ${}^{\triangleright}a$ for removing messages. The input and output labels of the alphabet composition are the non-shared input and output labels of the underlying alphabets.

Definition 2 (Alphabet composition). *Let* $\Sigma_{\mathcal{S}} = \mathsf{in}_{\mathcal{S}} \uplus \mathsf{out}_{\mathcal{S}} \uplus \mathsf{com}_{\mathcal{S}}$ *and* $\Sigma_{\mathcal{T}} = \mathsf{in}_{\mathcal{T}} \uplus \mathsf{out}_{\mathcal{T}} \uplus \mathsf{com}_{\mathcal{T}}$ *be two I/O-alphabets over channels* $C_{\mathcal{S}}$ *and* $C_{\mathcal{T}}$ *resp.* $\Sigma_{\mathcal{S}}$ *and* $\Sigma_{\mathcal{T}}$ *are* composable *if* $(\Sigma_{\mathcal{S}} \cup \Sigma_{\mathcal{T}}) \cap (C_{\mathcal{S}} \cup C_{\mathcal{T}}) = \emptyset$ *and* $\Sigma_{\mathcal{S}} \cap \Sigma_{\mathcal{T}} = (\mathsf{in}_{\mathcal{S}} \cap \mathsf{out}_{\mathcal{T}}) \cup (\mathsf{in}_{\mathcal{T}} \cap \mathsf{out}_{\mathcal{S}})$. *The composition of* $\Sigma_{\mathcal{S}}$ *and* $\Sigma_{\mathcal{T}}$ *is the I/O-alphabet* $\Sigma = \mathsf{in} \uplus \mathsf{out} \uplus \mathsf{com}$ *over the composed set of channels* $C = C_{\mathcal{S}} \uplus C_{\mathcal{T}} \uplus C_{\mathcal{ST}}$, *with new channels* $C_{\mathcal{ST}} = \Sigma_{\mathcal{S}} \cap \Sigma_{\mathcal{T}}$, *such that*

- $\mathsf{in} = (\mathsf{in}_{\mathcal{S}} \setminus \mathsf{out}_{\mathcal{T}}) \uplus (\mathsf{in}_{\mathcal{T}} \setminus \mathsf{out}_{\mathcal{S}})$,
- $\mathsf{out} = (\mathsf{out}_{\mathcal{S}} \setminus \mathsf{in}_{\mathcal{T}}) \uplus (\mathsf{out}_{\mathcal{T}} \setminus \mathsf{in}_{\mathcal{S}})$, *and*
- $\mathsf{com} = \{a^{\triangleright}, {}^{\triangleright}a \mid a \in C\}$[4] ♢

Two AIOPNs can be (asynchronously) composed, if their underlying I/O-alphabets are composable. The composition is constructed by taking the disjoint union of the underlying nets and adding a new channel place for each shared label. Every transition with shared output label a becomes a transition with the communication label $a^{\triangleright}$ that produces a token on the (new) channel place a and, similarly, any transition with shared input label a becomes a transition with the communication label ${}^{\triangleright}a$ that consumes a token from the (new) channel place a. For instance, the AIOPN $\mathcal{N}_3$ in Fig. 1c is the result of the asynchronous composition of the two AIOPNs $\mathcal{N}_1$ and $\mathcal{N}_2$ in Fig. 1a and Fig. 1b resp. The newly introduced channel place is the place msg.

Our approach looks very similar to open Petri nets, see e.g. [17], which use interface places for communication. But there are two important differences: First, we explicitely distinguish channel places thus being able to reason on the communication behaviour between composed components; see Sect. 4. The second difference is quite important from the software engineer's point of view. We do not use interface places to indicate communication abilities of a component but we use distinguished input and output labels instead. We believe that this has an important advantage to achieve separation of concerns: The designer of a component has not to take care whether the component will be used in a synchronous or in an asynchronous environment later on; this should be the decision of the system architect. Indeed open Petri nets already rely on asynchronous composition while our formalism would also support synchronous composition, see [19], and mixed architectures. Since synchronous composition relies on matching of transitions rather than communication channels we have not elaborated this case

[4] $\Sigma = \mathsf{in} \uplus \mathsf{out} \uplus \mathsf{com}$ is indeed a disjoint union, since for all $a \in C_{\mathcal{ST}}$ the communication labels $a^{\triangleright}$, ${}^{\triangleright}a$ are new names due to the general assumption that input and output labels are not of the form $x^{\triangleright}$, ${}^{\triangleright}x$.

here. The difference between AIOPNs and modal I/O-Petri nets introduced in [8] is that AIOPNs comprise distinguished channel places but they do not support modalities for refinement (yet).

Definition 3 (Asynchronous composition of AIOPNs). *Let* $\mathcal{N} = (C_\mathcal{N}, P_\mathcal{N}, T_\mathcal{N}, \Sigma_\mathcal{N}, W^-_\mathcal{N}, W^+_\mathcal{N}, \lambda_\mathcal{N}, m^0_\mathcal{N})$ *and* $\mathcal{M} = (C_\mathcal{M}, P_\mathcal{M}, T_\mathcal{M}, \Sigma_\mathcal{M}, W^-_\mathcal{M}, W^+_\mathcal{M}, \lambda_\mathcal{M}, m^0_\mathcal{M})$ *be two AIOPNs.* $\mathcal{N}$ *and* $\mathcal{M}$ *are* composable *if* $\Sigma_\mathcal{N}$ *and* $\Sigma_\mathcal{M}$ *are composable and if* $P_\mathcal{N} \cap P_\mathcal{M} = \emptyset$, $(P_\mathcal{N} \cup P_\mathcal{M}) \cap (\Sigma_\mathcal{N} \cap \Sigma_\mathcal{M}) = \emptyset$, *and* $T_\mathcal{N} \cap T_\mathcal{M} = \emptyset$. *In this case their* asynchronous composition *is the AIOPN* $\mathcal{N} \otimes_{pn} \mathcal{M} = (C, P, T, \Sigma, W^-, W^+, \lambda, m^0)$ *defined as follows:*

- $C = C_\mathcal{N} \uplus C_\mathcal{M} \uplus C_{\mathcal{NM}}$, *with* $C_{\mathcal{NM}} = \Sigma_\mathcal{N} \cap \Sigma_\mathcal{M}$,
- $P = P_\mathcal{N} \uplus P_\mathcal{M} \uplus C_{\mathcal{NM}}$,
- $T = T_\mathcal{N} \uplus T_\mathcal{M}$,
- Σ *is the alphabet composition of* $\Sigma_\mathcal{S}$ *and* $\Sigma_\mathcal{T}$,
- W^- *(resp.* W^+*) is the backward (forward) incidence matrix defined by:*

for all $p \in P_\mathcal{N} \cup P_\mathcal{M}$ *and* $t \in T$	*for all* $a \in C_{\mathcal{NM}}$ *and* $t \in T$
$W^-(p,t) = \begin{cases} W^-_\mathcal{N}(p,t) & \text{if } p \in P_\mathcal{N}, t \in T_\mathcal{N} \\ W^-_\mathcal{M}(p,t) & \text{if } p \in P_\mathcal{M}, t \in T_\mathcal{M} \\ 0 & \text{otherwise} \end{cases}$	$W^-(a,t) = \begin{cases} 1 & \text{if } \lambda(t) = {}^\triangleright a \\ 0 & \text{otherwise} \end{cases}$
$W^+(p,t) = \begin{cases} W^+_\mathcal{N}(p,t) & \text{if } p \in P_\mathcal{N}, t \in T_\mathcal{N} \\ W^+_\mathcal{M}(p,t) & \text{if } p \in P_\mathcal{M}, t \in T_\mathcal{M} \\ 0 & \text{otherwise} \end{cases}$	$W^+(a,t) = \begin{cases} 1 & \text{if } \lambda(t) = a^\triangleright \\ 0 & \text{otherwise} \end{cases}$

- $\lambda : T \to \Sigma$ *is defined, for all* $t \in T$, *by*

$$\lambda(t) = \begin{cases} \lambda_\mathcal{N}(t) & \text{if } t \in T_\mathcal{N},\ \lambda_\mathcal{N}(t) \notin \Sigma_\mathcal{N} \cap \Sigma_\mathcal{M} \\ \lambda_\mathcal{M}(t) & \text{if } t \in T_\mathcal{M},\ \lambda_\mathcal{M}(t) \notin \Sigma_\mathcal{N} \cap \Sigma_\mathcal{M} \\ {}^\triangleright\lambda_\mathcal{N}(t) & \text{if } t \in T_\mathcal{N},\ \lambda_\mathcal{N}(t) \in in_\mathcal{N} \cap out_\mathcal{M} \\ {}^\triangleright\lambda_\mathcal{M}(t) & \text{if } t \in T_\mathcal{M},\ \lambda_\mathcal{M}(t) \in in_\mathcal{M} \cap out_\mathcal{N} \\ \lambda_\mathcal{N}(t)^\triangleright & \text{if } t \in T_\mathcal{N},\ \lambda_\mathcal{N}(t) \in in_\mathcal{M} \cap out_\mathcal{N} \\ \lambda_\mathcal{M}(t)^\triangleright & \text{if } t \in T_\mathcal{M},\ \lambda_\mathcal{M}(t) \in in_\mathcal{N} \cap out_\mathcal{M} \end{cases}$$

- m^0 *is defined, for all* $p \in P$, *such that* $m^0(p) = m^0_\mathcal{N}(p)$ *if* $p \in P_\mathcal{N}$, $m^0(p) = m^0_\mathcal{M}(p)$ *if* $p \in P_\mathcal{M}$, *and* $m^0(p) = 0$ *otherwise.* ♢

3 Compositional Semantics

We extend the transition system semantics of labelled Petri nets defined in Sect. 2.1 to AIOPNs. For this purpose we introduce asynchronous I/O-transition systems which are labelled transition systems extended by channels and a *channel valuation* function $\text{val} : Q \longrightarrow \mathbb{N}^C$. The channel valuation function determines for each state $q \in Q$ how many messages are actually pending on each channel $a \in C$. For $q \in Q$ and $a \in C$ we use the notation $\text{val}(q)[a{+}{+}]$ to denote the updated map

$$\mathrm{val}(q)[a{+}{+}](x) = \begin{cases} \mathrm{val}(q)(a)+1 & \text{if } x = a, \\ \mathrm{val}(q)(x) & \text{otherwise.} \end{cases}$$

The updated map $\mathrm{val}(q)[a{-}{-}]$ is defined similarly. Instead of $\mathrm{val}(q)(a)$ we will often write $\mathrm{val}(q, a)$.

Definition 4 (Asynchronous I/O-transition system). *An* asynchronous I/O-transition system *(AIOTS) is a tuple* $\mathcal{S} = (C, \Sigma, Q, q^0, \longrightarrow, \mathrm{val})$*, such that*

- $(\Sigma, Q, q^0, \longrightarrow)$ *is a labelled transition system,*
- C *is a finite set of channels,*
- $\Sigma = \mathsf{in} \uplus \mathsf{out} \uplus \mathsf{com}$ *is an I/O-alphabet over* C*,*
- $\mathrm{val} : Q \longrightarrow \mathbb{N}^C$ *is a function, such that for all* $a \in C, q, q' \in Q$*:*
  - $\mathrm{val}(q^0, a) = 0$*,*
  - $q \xrightarrow{a^{\triangleright}} q' \implies \mathrm{val}(q') = \mathrm{val}(q)[a{+}{+}]$*,*
  - $q \xrightarrow{{}^{\triangleright}a} q' \implies \mathrm{val}(q, a) > 0$ *and* $\mathrm{val}(q') = \mathrm{val}(q)[a{-}{-}]$*, and*
  - *for all* $x \in (\mathsf{in} \cup \mathsf{out}), q \xrightarrow{x} q' \implies \mathrm{val}(q') = \mathrm{val}(q)$*.* ♢

The first condition for val assumes that initially all communication channels are empty. The second condition states that transitions with labels $a^{\triangleright}$ and ${}^{\triangleright}a$ have the desired effect of putting one message on a channel (consuming one message from a channel resp.). The last condition requires that the input and output actions of an open system do not change the valuation of any channel. Sometimes we need to reason about the number of messages on a subset $B \subseteq C$ of the channels in a state $q \in Q$. We define $\mathrm{val}(q, B) = \sum_{a \in B} \mathrm{val}(q, a)$.

The semantics of an asynchronous I/O-Petri net $\mathcal{N}$ is given by its *associated* asynchronous I/O-transition system $\mathsf{aiots}(\mathcal{N})$. It is based on the transition system semantics of a labelled Petri net (see Sect. 2.1) such that markings become states, but additionally we define the valuation of a channel in a current state m by the number of tokens on the channel under the marking m.

Definition 5 (Associated asynchronous I/O-transition system). *Let* $\mathcal{N} = (C, P, T, \Sigma, W^-, W^+, \lambda, m^0)$ *be an AIOPN. The AIOTS* associated with $\mathcal{N}$ *is given by* $\mathsf{aiots}(\mathcal{N}) = (C, \Sigma, Q, q^0, \longrightarrow, \mathrm{val})$*, such that*

- $(\Sigma, Q, q^0, \longrightarrow) = \mathsf{lts}(P, T, \Sigma, W^-, W^+, \lambda, m^0)$*,*
- *for all* $a \in C$ *and* $m \in Q, \mathrm{val}(m, a) = m(a)$*.* ♢

Example 6. The transition systems associated with the AIOPNs $\mathcal{N}_1$ and $\mathcal{N}_2$ in Fig. 1a and 1b have two reachable states and the transitions between them correspond directly to their Petri net representations. The situation is different for the AIOPN $\mathcal{N}_3$ in Fig. 1c. It has infinitly many reachable markings and hence its associated AIOTS has infinitely many reachable states. Fig. 2 shows an excerpt of it. The states indicate the number of tokens in each place in the order p_0, p_1, msg, p_2, p_3. The initial state is underlined.

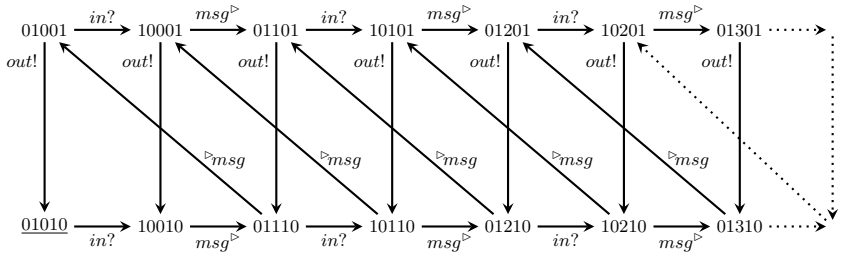


Fig. 2: Part of the associated AIOTS for $\mathcal{N}_3$ in Fig. 1c.

Like AIOPNs also two AIOTSs can be asynchronously composed, if their underlying I/O-alphabets are composable. The composition is constructed by introducing a new communication channel for each shared input/output action and by appropriate transitions for the corresponding communication actions that modify the valuation of the new channels (see items 3 and 4 in Def. 7). Since the states of the composition must record the number of messages on the new channels $C_{\mathcal{ST}}$, the state space of the composition adds to the Cartesian product of the underlying state spaces the set $\mathbb{N}^{C_{\mathcal{ST}}}$ of valuations of the new channels. For a valuation $\boldsymbol{v} : C_{\mathcal{ST}} \mapsto \mathbb{N}$ and channel $a \in C_{\mathcal{ST}}$ we use the notation $\boldsymbol{v}[a{+}{+}]$ ($\boldsymbol{v}[a{-}{-}]$ resp.) to denote the updated map which increments (decrements) the value of a by 1 and leaves the values of all other channels unchanged.

Definition 7 (Asynchronous composition of AIOTS).
Let $\mathcal{S} = (C_{\mathcal{S}}, \Sigma_{\mathcal{S}}, Q_{\mathcal{S}}, q^0_{\mathcal{S}}, \longrightarrow_{\mathcal{S}}, \mathrm{val}_{\mathcal{S}})$ *and* $\mathcal{T} = (C_{\mathcal{T}}, \Sigma_{\mathcal{T}}, Q_{\mathcal{T}}, q^0_{\mathcal{T}}, \longrightarrow_{\mathcal{T}}, \mathrm{val}_{\mathcal{T}})$ *be two AIOTSs.* $\mathcal{S}$ *and* $\mathcal{T}$ *are composable if* $\Sigma_{\mathcal{S}}$ *and* $\Sigma_{\mathcal{T}}$ *are composable. In this case their asynchronous composition is the AIOTS* $\mathcal{S} \otimes \mathcal{T} = (C, \Sigma, Q, q^0, \longrightarrow, \mathrm{val})$ *defined as follows:*

- $C = C_{\mathcal{S}} \uplus C_{\mathcal{T}} \uplus C_{\mathcal{ST}}$, *with* $C_{\mathcal{ST}} = \Sigma_{\mathcal{S}} \cap \Sigma_{\mathcal{T}}$,
- Σ *is the alphabet composition of* $\Sigma_{\mathcal{S}}$ *and* $\Sigma_{\mathcal{T}}$,
- $Q \subseteq Q_{\mathcal{S}} \times Q_{\mathcal{T}} \times \mathbb{N}^{C_{\mathcal{ST}}}$,
- $q^0 = (q^0_{\mathcal{S}}, q^0_{\mathcal{T}}, \mathbf{0}) \in Q$, *with* $\mathbf{0}$ *being the zero-map,*
- Q *and* $\longrightarrow$ *are inductively defined as follows whenever* $(q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \in Q$*:*
  1: *For all* $a \in (\Sigma_{\mathcal{S}} \setminus C_{\mathcal{ST}})$, *if* $q_{\mathcal{S}} \xrightarrow{a}_{\mathcal{S}} q'_{\mathcal{S}}$ *then* $(q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \xrightarrow{a} (q'_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v})$ *and* $(q'_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \in Q$.
  2: *For all* $a \in (\Sigma_{\mathcal{T}} \setminus C_{\mathcal{ST}})$, *if* $q_{\mathcal{T}} \xrightarrow{a}_{\mathcal{T}} q'_{\mathcal{T}}$ *then* $(q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \xrightarrow{a} (q_{\mathcal{S}}, q'_{\mathcal{T}}, \boldsymbol{v})$ *and* $(q_{\mathcal{S}}, q'_{\mathcal{T}}, \boldsymbol{v}) \in Q$.
  3: *For all* $a \in \mathsf{in}_{\mathcal{S}} \cap \mathsf{out}_{\mathcal{T}}$,
    3.1: *if* $q_{\mathcal{S}} \xrightarrow{a}_{\mathcal{S}} q'_{\mathcal{S}}$ *and* $\boldsymbol{v}(a) > 0$ *then* $(q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \xrightarrow{\triangleright a} (q'_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}[a{-}{-}])$ *and* $(q'_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}[a{-}{-}]) \in Q$,
    3.2: *if* $q_{\mathcal{T}} \xrightarrow{a}_{\mathcal{T}} q'_{\mathcal{T}}$ *then* $(q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \xrightarrow{a^{\triangleright}} (q_{\mathcal{S}}, q'_{\mathcal{T}}, \boldsymbol{v}[a{+}{+}])$ *and* $(q_{\mathcal{S}}, q'_{\mathcal{T}}, \boldsymbol{v}[a{+}{+}]) \in Q$.
  4: *For all* $a \in \mathsf{in}_{\mathcal{T}} \cap \mathsf{out}_{\mathcal{S}}$,

4.1: if $q_{\mathcal{S}} \xrightarrow{a}_{\mathcal{S}} q'_{\mathcal{S}}$ *then* $(q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \xrightarrow{a^{\triangleright}} (q'_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}[a{+}{+}])$ *and* $(q'_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}[a{+}{+}]) \in Q$,

4.2: if $q_{\mathcal{T}} \xrightarrow{a}_{\mathcal{T}} q'_{\mathcal{T}}$ *and* $\boldsymbol{v}(a) > 0$ *then* $(q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \xrightarrow{{}^{\triangleright}a} (q_{\mathcal{S}}, q'_{\mathcal{T}}, \boldsymbol{v}[a{-}{-}])$ *and* $(q_{\mathcal{S}}, q'_{\mathcal{T}}, \boldsymbol{v}[a{-}{-}]) \in Q$.

– *For all* $(q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \in Q$ *and* $a \in C$,

$$\mathrm{val}((q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}), a) = \begin{cases} \mathrm{val}_{\mathcal{S}}(q_{\mathcal{S}}, a) & \text{if } a \in C_{\mathcal{S}} \\ \mathrm{val}_{\mathcal{T}}(q_{\mathcal{T}}, a) & \text{if } a \in C_{\mathcal{T}} \\ \boldsymbol{v}(a) & \text{if } a \in C_{\mathcal{S}\mathcal{T}} \end{cases}$$

For the rules (1),(3.1) and (4.1), we say that the resulting transition in the composition is triggered *by* $\mathcal{S}$. *Let* ρ *be a trace of* $\mathcal{S} \otimes \mathcal{T}$ *starting from a state* $q = (q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \in Q$. *The* projection *of* ρ *to* $\mathcal{S}$, *denoted by* $\rho|_{\mathcal{S}}$, *is the sequence of transitions of* $\mathcal{S}$, *starting from* $q_{\mathcal{S}}$, *which have triggered corresponding transitions in* ρ. ♢

The following theorem shows that the transition system semantics of asynchronous I/O-Petri nets is compositional. The proof is given in [12].

Theorem 8. *Let* $\mathcal{N}$ *and* $\mathcal{M}$ *be two composable AIOPNs. Then it holds that* $\mathsf{aiots}(\mathcal{N} \otimes_{pn} \mathcal{M}) = \mathsf{aiots}(\mathcal{N}) \otimes \mathsf{aiots}(\mathcal{M})$ *(up to bijection between state spaces).*

4 Channel Properties and Their Compositionality

In this section we consider various properties concerning the asynchronous communication via channels. We give a classification of the properties, show their relationships and prove that they are compositional w.r.t. asynchronous composition, a prerequisite for incremental design.

4.1 Channel Properties

We consider two classes of channel properties. The first class deals with the requirements that messages sent to a communication channel should also be consumed; the second class concerns the termination of communication in the sense that if consumption from a channel has been stopped then also production on this channel will stop. The channel properties will be defined by considering the semantics of AIOPNs, i.e. they will be formulated for AIOTSs.

Some of the properties, precisely the "necessarily properties" of type (c) in Def. 10 below, rely on the consideration of system runs. In principle a system run is a maximal execution trace; it can be infinite but also finite if no further actions are enabled. It is important to remember, that we deal with open systems whose possible behaviours are also determined by the environment. Hence, the definition of a system run must take into account the possibility that the system may stop in a state where the environment does not serve any offered input of the system while at the same time the system has no enabled autonomous action, i.e.

an action which is not an input from the environment. Such states will be called pure input states. They correspond to markings that "stop except for inputs" in [24]. Note that all possible communication actions inside the system can be autonomously executed. The same holds for output actions to the environment, since we are working with asynchronous communication such that messages can always be sent, even if they are never accepted by the environment. Formally, system runs are defined as follows.

Let $\mathcal{S} = (C, \Sigma, Q, q^0, \longrightarrow, \mathrm{val})$ be an AIOTS with $\Sigma = \mathsf{in} \uplus \mathsf{out} \uplus \mathsf{com}$. A state $q \in Q$ is called a *pure input state* if $\mathrm{Post}(q, \Sigma \setminus \mathsf{in}) = \emptyset$, i.e. only inputs are enabled. A pure input state is a potential deadlock, as the environment of $\mathcal{S}$ might not serve any inputs for $\mathcal{S}$. Let $q_1 \in Q$. A *run* of $\mathcal{S}$ starting in q_1 is a trace of $\mathcal{S}$ starting in q_1, that is either infinite or finite such that its last state is a pure input state. We denote the set of all runs of $\mathcal{S}$ starting from q_1 as $\mathrm{run}_{\mathcal{S}}(q_1)$.

In the following we also assume that system runs are only executed in a runtime infrastructure which follows a weakly fair scheduling policy. In our context this means that any autonomous action a, that is always enabled from a certain state on, will infinitely often be executed. Formally, a run $\rho \in \mathrm{run}_{\mathcal{S}}(q_1)$ with $q_1 \in Q, \rho = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots$, is called *weakly fair* if it is finite or if it is infinite and for all $a \in (\Sigma \setminus \mathsf{in})$ the following holds:

$$(\exists k \geq 1 \,.\, \forall i \geq k \,.\, q_i \xrightarrow{a}) \implies (\forall k \geq 1 \,.\, \exists i \geq k \,.\, a_i = a).$$

We denote the set of all weakly fair runs of $\mathcal{S}$ starting from q_1 by $\mathrm{wfrun}_{\mathcal{S}}(q_1)$. It should be noted that for our results it is sufficient to use weak fairness instead of strong fairness.[5]

Example 9. Let $\mathcal{S} = \mathsf{aiots}(\mathcal{N}_3)$ be the associated AIOTS of the Petri net $\mathcal{N}_3$ in Fig. 1c. An excerpt of $\mathcal{S}$ has been shown in Fig. 2. The following are three traces of $\mathcal{S}$ starting in the initial state 01010:

$\rho_0 = 01010,$
$\rho_1 = 01010 \xrightarrow{in?} 10010 \xrightarrow{msg^{\triangleright}} 01110 \xrightarrow{{}^{\triangleright}msg} 01001,$
$\rho_2 = 01010 \xrightarrow{in?} 10010 \xrightarrow{msg^{\triangleright}} 01110 \xrightarrow{{}^{\triangleright}msg} 01001 \xrightarrow{out!} 01010.$

The traces ρ_0 and ρ_2 are runs of $\mathcal{S}$ while ρ_1 is not a run, since 01001 is not a pure input state. Now consider the infinite trace indicated at the bottom line in Fig. 2, that is an infinite alternation of $in?$ and $msg^{\triangleright}$. The trace is a run, since it is infinite. But the run is not weakly fair, since from the second state on ${}^{\triangleright}msg$ is always enabled but never taken.

Our first class of channel properties deals with the consumption of previously produced messages. We consider four groups of such properties (P1) - (P4) with different strength. In each case we consider three variants which all are parametrised w.r.t. a subset B of the communication channels.

Let us discuss the consuming properties (P1) of Def. 10 below for an AIOTS $\mathcal{S}$ and a subset B of its channels. Property (P1.a) requires, for each channel

[5] For a discussion of the different fairness properties see, e.g., [3].

$a \in B$, that if in an arbitrary reachable state q of $\mathcal{S}$ there is a message available on a, then $\mathcal{S}$ can consume the message possibly after the execution of some autonomous actions. Let us comment on the role of the environment for the formulation of this property. First, we consider arbitrary reachable states $q \in \mathrm{Post}^*(q^0)$ with q^0 being the initial state of $\mathcal{S}$. This means that we take into account the worst environment which can let $\mathcal{S}$ go everywhere by providing (nondeterministically) all inputs that $\mathcal{S}$ can accept. Then, at some point at which a message is available on channel a, the environment can stop to provide further inputs and waits whether $\mathcal{S}$ can *autonomously* reach a state $q' \in \mathrm{Post}^*(q, \Sigma \setminus \mathsf{in})$ in which it can consume from a, i.e. execute ${}^{\triangleright}a$. To allow autonomous actions before consumption is inspired by the property of "output compatibility" studied for synchronously composed transition systems in [14]. Property (P1.b) does not allow autonomous actions before consumption. It requires that $\mathcal{S}$ can immediately consume the message in state q, similar to the property of specified reception in [5]. Property (P1.c) requires that the message will definitely be consumed on each weakly fair run of $\mathcal{S}$ starting from q and, due to the definition of a system run, that this will happen in any environment whatever inputs are provided.

As an example consider the AIOTS $\mathcal{S} = \mathsf{aiots}(\mathcal{N}_3)$ associated with the AIOPN $\mathcal{N}_3$ in Fig. 1c and its reachable state 01101 such that one message is on channel msg. In this state $\mathcal{S}$ can autonomously perform the output $out!$ reaching state 01110 and then it can consume the message by performing ${}^{\triangleright}msg$. Since also in all other reachable states in which the channel is not empty $\mathcal{S}$ can autonomously reach a state in which it can consume from the channel, $\mathcal{S}$ satisfies property (P1.a) (for its only channel msg). However, $\mathcal{S}$ is not strongly consuming (P1.b). For instance in state 01101, $\mathcal{S}$ cannot immediately consume the message. On the other hand, $\mathcal{S}$ is necessarily consuming (P1.c). Whenever in a reachable state q the channel is not empty an autonomous action, either ${}^{\triangleright}msg$ or $out!$, is enabled. Hence q is not a pure input state and, due to the weak fairness condition, eventually ${}^{\triangleright}msg$ or $out!$ must be performed in any weakly fair run starting from q. If ${}^{\triangleright}msg$ is performed we are done. If $out!$ is performed we reach a state where ${}^{\triangleright}msg$ is enabled and with the same reasoning eventually ${}^{\triangleright}msg$ will be performed. This can be easily detected by considering Fig. 2.

The other groups of properties (P2) - (P4) express successively stronger (or equivalent) requirements on the kind of consumption. For instance, (P3) requires that the consumption will lead to a state in which the channel is empty. Again we distinguish if this can be achieved after some autonomous actions (P3.a), can be achieved immediately (P3.b), or must be achieved in any weakly fair run (P3.c).

Definition 10 (Consumption properties). *Let* $\mathcal{S} = (C, \Sigma, Q, q^0, \longrightarrow, \mathrm{val})$ *be an AIOTS with I/O-alphabet* $\Sigma = \mathsf{in} \uplus \mathsf{out} \uplus \mathsf{com}$ *and let* $B \subseteq C$ *be a subset of its channels.*

P1: (Consuming)

a) $\mathcal{S}$ *is* B-consuming*, if for all* $a \in B$ *and all* $q \in \mathrm{Post}^*(q^0)$*,*

$$\mathrm{val}(q, a) > 0 \implies \exists q' \in \mathrm{Post}^*(q, \Sigma \setminus \mathsf{in}) \,.\, q' \xrightarrow{{}^{\triangleright}a} .$$

b) $\mathcal{S}$ *is* strongly B-consuming*, if for all* $a \in B$ *and all* $q \in \mathrm{Post}^*(q^0)$*,*
$\mathrm{val}(q,a) > 0 \implies q \xrightarrow{{}^{\triangleright}a}$.

c) $\mathcal{S}$ *is* necessarily B-consuming*, if for all* $a \in B$ *and all* $q \in \mathrm{Post}^*(q^0)$*,*
$\mathrm{val}(q,a) > 0 \implies \forall \rho \in \mathrm{wfrun}_{\mathcal{S}}(q) \,.\, {}^{\triangleright}a \in \rho$.

P2: (Decreasing)

a) $\mathcal{S}$ *is* B-decreasing*, if for all* $a \in B$ *and all* $q \in \mathrm{Post}^*(q^0)$*,*
$\mathrm{val}(q,a) > 0 \implies \exists q' \in \mathrm{Post}^*(q, \Sigma \setminus \mathsf{in}) \,.\, \mathrm{val}(q',a) < val(q,a)$.

b) $\mathcal{S}$ *is* strongly B-decreasing*, if for all* $a \in B$ *and all* $q \in \mathrm{Post}^*(q^0)$*,*
$\mathrm{val}(q,a) > 0 \implies \exists q' \in \mathrm{Post}(q, \Sigma \setminus \mathsf{in}) \,.\, \mathrm{val}(q',a) < val(q,a)$.

c) $\mathcal{S}$ *is* necessarily B-decreasing*, if for all* $a \in B$ *and all* $q \in \mathrm{Post}^*(q^0)$*,*
$\mathrm{val}(q,a) > 0 \implies \forall \rho \in \mathrm{wfrun}_{\mathcal{S}}(q),\ \exists q' \in \rho \,.\, \mathrm{val}(q',a) < \mathrm{val}(q,a)$.

P3: (Emptying)

a) $\mathcal{S}$ *is* B-emptying*, if for all* $a \in B$ *and all* $q \in \mathrm{Post}^*(q^0)$*,*
$\mathrm{val}(q,a) > 0 \implies \exists q' \in \mathrm{Post}^*(q, \Sigma \setminus \mathsf{in}) \,.\, \mathrm{val}(q',a) = 0$.

b) $\mathcal{S}$ *is* strongly B-emptying*, if for all* $a \in B$ *and all* $q \in \mathrm{Post}^*(q^0)$*,*
$\mathrm{val}(q,a) > 0 \implies \exists q' \in \mathrm{Post}(q, \Sigma \setminus \mathsf{in}) \,.\, \mathrm{val}(q',a) = 0$.

c) $\mathcal{S}$ *is* B-necessarily emptying*, if for all* $a \in B$ *and all* $q \in \mathrm{Post}^*(q^0)$*,*
$\mathrm{val}(q,a) > 0 \implies \forall \rho \in \mathrm{wfrun}_{\mathcal{S}}(q),\ \exists q' \in \rho \,.\, \mathrm{val}(q',a) = 0$.

P4: (Wholly emptying)

a) $\mathcal{S}$ *is* B-wholly emptying*, if for all* $q \in \mathrm{Post}^*(q^0)$*,*
$\mathrm{val}(q,B) > 0 \implies \exists q' \in \mathrm{Post}^*(q, \Sigma \setminus \mathsf{in}) \,.\, \mathrm{val}(q',B) = 0.$

b) $\mathcal{S}$ *is* strongly B-wholly emptying*, if for all* $q \in \mathrm{Post}^*(q^0)$*,*
$\mathrm{val}(q,B) > 0 \implies \exists q' \in \mathrm{Post}(q, \Sigma \setminus \mathsf{in}) \,.\, \mathrm{val}(q',B) = 0.$

c) $\mathcal{S}$ *is* B-necessarily wholly emptying*, if for all* $q \in \mathrm{Post}^*(q^0)$*,*
$\mathrm{val}(q,B) > 0 \implies \forall \rho \in \mathrm{wfrun}_{\mathcal{S}}(q),\ \exists q' \in \rho \,.\, \mathrm{val}(q',B) = 0$. ♢

Note if the initial state of $\mathcal{S}$ is reachable from all other reachable states, i.e. the initial state is a *home state*, then $\mathcal{S}$ is B-wholly emptying.

The next class of channel properties concerns the termination of communication. We consider two variants: (P5.a) requires that in any weakly fair run, in which consumption from a channel a has stopped, only finitely many subsequent productions are possible, i.e. the channel is closed after a while. Property (P5.b) expresses that the channel is immediately closed.

Definition 11 (Communication stopping). *Let* $\mathcal{S}$ *be an AIOTS and* $B \subseteq C$ *be a subset of its channels.*

P5: (Communication stopping)

a) $\mathcal{S}$ *is* B-communication stopping*, if for all* $q \in \mathrm{Post}^*(q^0),\ \rho \in \mathrm{wfrun}_{\mathcal{S}}(q)$ *and* $a \in B$*,* $\sharp_\rho({}^{\triangleright}a) = 0 \implies \sharp_\rho(a^{\triangleright}) < \infty$.

b) $\mathcal{S}$ *is* strongly B-communication stopping*, if for all* $q \in \mathrm{Post}^*(q^0),\ \rho \in \mathrm{wfrun}_{\mathcal{S}}(q)$ *and* $a \in B$*,* $\sharp_\rho({}^{\triangleright}a) = 0 \implies \sharp_\rho(a^{\triangleright}) = 0$. ♢

We say that an AIOTS $\mathcal{S}$ has a channel property P, if $\mathcal{S}$ has property P with respect to the set C of all channels of $\mathcal{S}$.

Definition 12 (Channel properties of AIOPNs). *Let P be an arbitrary channel property as defined above. An AIOPN $\mathcal{N}$ has property P w.r.t. some subset B of its channels if its generated AIOTS* $\mathsf{aiots}(\mathcal{N})$ *has property P w.r.t. B; $\mathcal{N}$ has property P if* $\mathsf{aiots}(\mathcal{N})$ *has property P.*

Relevance of channel properties. The generic properties that we have defined fit well with properties related to specific distributed mechanisms, algorithms and applications. For instance:

- When sending an email the user must be confident that its mail will be eventually read. Such a property can be formalized as the necessarily consuming property.
- Most distributed applications can be designed with an underlying token circulation between the processes of the applications. This requires that at any time there is at most one token in all channels and that this token can be immediately handled. Such a property can be formalized as the strong wholly emptying property.
- Recovery points are useful for applications prone to faults. While algorithms for building recovery points can handle non-empty channels, the existence (and identification) of states with empty channels eases this task. Such a property can be formalized as the necessarily wholly emptying property.
- In UNIX, one often requires that a process should not write in a socket when no reader of the socket is still present (and this could raise a signal). Such a property can be formalized as the strong communication stopping property.

4.2 Relationships Between Channel Properties

Table 1 shows relationships between the channel properties and pointers to examples of AIOPNs from Fig. 1 and Fig. 3 which have the indicated properties.

All the downward implications inside the boxes are direct consequences of the definitions. It is trivial to see that downward implication 3 is an equivalence, since *immediate* consumption leads to a decreasing valuation. Downward implications 9 and 16 are equivalences, since repeated decreasing of messages on a channel will eventually lead to an empty channel. The implications 4, 5, 7, 11-14 and 18 are proved in [12]. Additionally we have that all properties in box b) of Tab. 1 imply the strongest property in box a), since if $\mathcal{S}$ is strongly B-consuming we can by repeated consumption empty all channels in B.

Let us now discuss some counterexamples. As discussed in Sect. 4.1, a counterexample for the converse of implication 7 is the AIOPN $\mathcal{N}_3$ in Fig. 1c. An obvious counterexample for the converse of the implications 2, 10, 11, 12, 13 is given by the AIOPN $\mathcal{N}_4$ shown in Fig. 3a. $\mathcal{N}_4$ is also a counterexample for implication 6. The AIOPN $\mathcal{N}_5$ in Fig. 3b with channels a and b is a counterexample for the converse of implication 15. The net can empty each single channel a and b but it can never have both channels empty at the same time (after the first message has been produced on a channel). A counterexample for the converse of implication 14 is shown by the net $\mathcal{N}_6$ in Fig. 3c. The net can put a token on

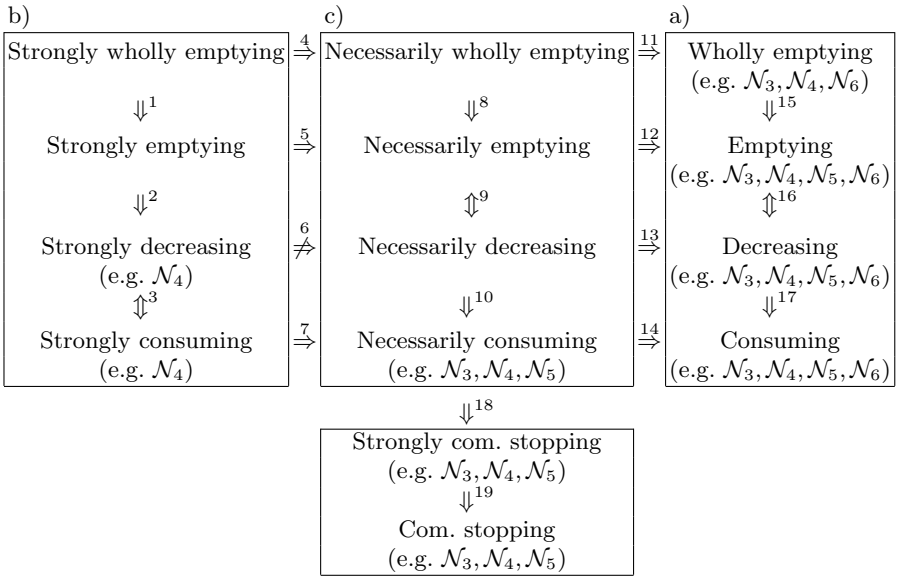


Table 1: Relationships between channel properties and examples.

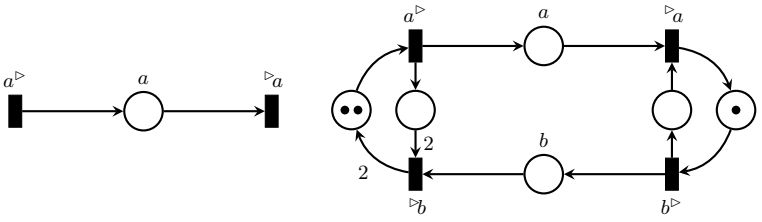
the channel a, but afterwards the transition ${}^{\triangleright}a$ is not necessarily always enabled which means there exists a weakly fair run such that there is always a token in a and ${}^{\triangleright}a$ is never fired.

Counterexamples for the converse of implication 17 rely on the idea to produce twice while consuming once. A counterexample for the converse of implication 18 is provided by a net that first produces a finite number n of messages on a channel, then it consumes less than n of these messages and then it stops. Counterexamples for the remaining converse implications are straightforward to construct.

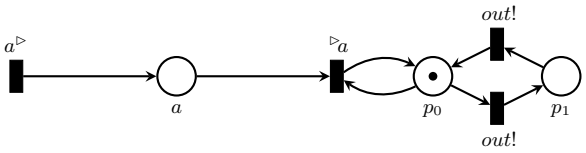
4.3 Compositionality of Channel Properties

Modular verification of systems is an important goal in any development method. In our context this concerns the question whether channel properties are preserved in arbitrary environments or, more precisely, whether they are preserved under asynchronous composition. In this section we show that indeed all channel properties defined above are compositional. This can be utilised to get a method for incremental design. The proofs of the results of this section are given in [12].

In order to relate channel properties of asynchronous compositions to channel properties of their constituent parts we need the next two lemmas. The first one shows that autonomous executions of constituent parts (not involving inputs) can be lifted to executions of compositions. This is the essence to prove compositionality of the properties of type (a) and type (b) in Def. 10.



(a) $\mathcal{N}_4$ (b) $\mathcal{N}_5$



(c) $\mathcal{N}_6$

Fig. 3: Examples of AIOPNs.

Lemma 13. *Let* $\mathcal{S} = (C_{\mathcal{S}}, \Sigma_{\mathcal{S}}, Q_{\mathcal{S}}, q^0_{\mathcal{S}}, \longrightarrow_{\mathcal{S}}, \mathrm{val}_{\mathcal{S}}), \mathcal{T} = (C_{\mathcal{T}}, \Sigma_{\mathcal{T}}, Q_{\mathcal{T}}, q^0_{\mathcal{T}}, \longrightarrow_{\mathcal{T}}, \mathrm{val}_{\mathcal{T}})$ *be two composable AIOTSs, and let* $\mathcal{S} \otimes \mathcal{T} = (C, \Sigma, Q, q^0, \longrightarrow, \mathrm{val})$. *For all* $(q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \in \mathrm{Post}^*(q^0)$ *and* $\sigma \in (\Sigma_{\mathcal{S}} \setminus \mathsf{in}_{\mathcal{S}})^*$ *it holds that*

$$q_{\mathcal{S}} \xrightarrow{\sigma}_{\mathcal{S}} q'_{\mathcal{S}} \implies \exists \boldsymbol{v'} \,.\, (q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \xrightarrow{\bar{\sigma}} (q'_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v'}),$$

with $\bar{\sigma} \in (\Sigma \setminus \mathsf{in})^*$ *obtained from* σ *by replacing any occurrence of a shared label* $a \in \mathsf{out}_{\mathcal{S}} \cap \mathsf{in}_{\mathcal{T}}$ *by the communication label* $a^{\triangleright}$.

The next lemma is crucial to prove compositionality of the "necessarily" properties of type (c) in Def. 10 and the communication stopping properties in Def. 11. It shows that projections of weakly fair runs are weakly fair runs again. This result can only be achieved in the asynchronous context.

Lemma 14. *Let* $\mathcal{S}$, $\mathcal{T}$ *be two composable AIOTSs, and* $\mathcal{S} \otimes \mathcal{T} = (C, \Sigma, Q, q^0, \longrightarrow, \mathrm{val})$. *Let* $q = (q_{\mathcal{S}}, q_{\mathcal{T}}, \boldsymbol{v}) \in Q$ *and* $\rho \in \mathrm{wfrun}_{\mathcal{S}\otimes\mathcal{T}}(q)$ *be a weakly fair run. Then* $\rho|_{\mathcal{S}} \in \mathrm{wfrun}_{\mathcal{S}}(q_{\mathcal{S}})$, *is a weakly fair run.*

Proposition 15 (Compositionality of Channel Properties). *Let* $\mathcal{S}$ *and* $\mathcal{T}$ *be two composable AIOTSs such that* $C_{\mathcal{S}}$ *is the set of channels of* $\mathcal{S}$. *Let* $B \subseteq C_{\mathcal{S}}$ *and let* P *be an arbitrary channel property as defined in Sec. 4.1. If* $\mathcal{S}$ *has property* P *with respect to the channels* B, *then* $\mathcal{S} \otimes \mathcal{T}$ *has property* P *with respect to the channels* B. *This holds analogously for asynchronous I/O-Petri nets (due to the compositional semantics of AIOPNs; see Thm. 8).*

Proposition 15 leads to the desired modular verification result for all properties except wholly emptying (P4): In order to check that a composition $\mathcal{N} \otimes_{pn} \mathcal{M}$ of two AIOPNs has a channel property P, i.e. P holds for all channels of the composition, it is sufficient to know that $\mathcal{N}$ and $\mathcal{M}$ have property P and to prove that $\mathcal{N} \otimes_{pn} \mathcal{M}$ has property P with respect to the new channels introduced by the asynchronous composition.

Theorem 16 (Incremental Design). *Let $\mathcal{N}$ and $\mathcal{M}$ be two composable AIOPNs with shared actions $\Sigma_{\mathcal{N}} \cap \Sigma_{\mathcal{M}}$ and let P be an arbitrary channel property but (P4). If both $\mathcal{N}$ and $\mathcal{M}$ have property P and if $\mathcal{N} \otimes_{pn} \mathcal{M}$ has property P with respect to the new channels $\Sigma_{\mathcal{N}} \cap \Sigma_{\mathcal{M}}$, then $\mathcal{N} \otimes_{pn} \mathcal{M}$ has property P.*

5 Decidability of Channel Properties

We begin this section by recalling some information related to semi-linear sets and decision procedures in Petri nets that we use in our proofs.

Let $E \subseteq \mathbb{N}^k$, E is a *linear set* if there exists a finite set of vectors of $\mathbb{N}^k$ $\{v_0, \ldots, v_n\}$ such that $E = \{v_0 + \sum_{1 \leq i \leq n} \lambda_i v_i \mid \forall i \ \lambda_i \in \mathbb{N}\}$. A *semi-linear set* [10] is a finite union of linear sets; a representation of it is given by the family of finite sets of vectors defining the corresponding linear sets. Semi-linear sets are *effectively* closed w.r.t. union, intersection and complementation. This means that one can compute a representation of the union, intersection and complementation starting from a representation of the original semi-linear sets. E is an *upward closed set* if $\forall v \in E.\ v' \geq v \Rightarrow v' \in E$. An upward closed set has a finite set of minimal vectors denoted $\min(E)$. An upward closed set is a semi-linear set which has a representation that can be derived from the equation $E = \min(E) + \mathbb{N}^k$ if $\min(E)$ is computable.

Given a Petri net $\mathcal{N}$ and a marking m, the *reachability* problem consists in deciding whether m is reachable from m_0 in $\mathcal{N}$. This problem is decidable [18] but none of the associated algorithms are primitive recursive. Furthermore this procedure can be adapted to semi-linear sets when markings are identified to vectors of $\mathbb{N}^{|P|}$. Based on reachability analysis, the authors of [9] design an algorithm that decides whether a marking m is a *home state*, i.e. m is reachable from any reachable marking. A more general problem is in fact decidable: given a subset of places P' and a (sub)marking m on this subset, is it possible from any reachable marking to reach a marking that coincides on P' with m?

In [20], the *coverability* is shown to be EXPSPACE-complete. The coverability problem consists in determining, given a net and a target marking, whether one can reach a marking greater or equal than the target. In [26] given a Petri net, several procedures have been designed to compute the minimal set of markings of several interesting upward closed sets. In particular, given an upward closed set $Target$, by a backward analysis one can compute the (representation of) upward closed set from which $Target$ is reachable. Using the results of [20], this algorithm performs in EXPSPACE.

While in Petri nets, strong fairness is undecidable [6], weak fairness is decidable and more generally, the existence of an infinite sequence fulfilling a formula of the following fragment of LTL is decidable [15]. The literals are (1) comparisons between places markings and values, (2) transition firings and (3) their negations. Formulas are inductively defined as literals, conjunction or disjunction of formulas and $GF\varphi$ where GF is the infinitely often operator and φ is a formula.

The next theorem establishes the decidability of the strong properties of type (b) of Def. 10. Observe that their proofs given in [12] are closely related and that they rely on the decidability of reachability and coverability problems.

Theorem 17. *The following problems are decidable for AIOPNs: Is an AIOPN $\mathcal{N}$ strongly B-consuming, strongly B-decreasing, strongly B-emptying, strongly B-wholly emptying?*

The next theorem establishes the decidability of the properties of type (a) of Def. 10. Observe that their proofs rely on (1) the effectiveness of backward analysis for upward closed marking sets (2) the decidability of reachability and home space problems and (3) appropriate transformations of the net.

Theorem 18. *The following problems are decidable for AIOPNs: Is an AIOPN $\mathcal{N}$ B-consuming, B-decreasing, B-emptying, B-wholly emptying?*

Proof.
B-consuming. Given an AIOPN $\mathcal{N}$ and B a subset of its channels, one decides whether $\mathcal{N}$ is B-consuming as follows.

Let $a \in B$ and E_a be the upward closed set of markings defined by:

$$E_a = \{m \mid \exists t \in T \text{ with } \lambda(t) = {}^{\triangleright}a \text{ and } m \geq W^-(t)\}$$

E_a is the set of markings from which one can immediately consume some message a. Let F_a be the upward closed set of markings defined by:

$$F_a = \{m \mid \exists m' \in E_a \ \exists \sigma \in T^*.\ \lambda(\sigma) \in (\Sigma \setminus \mathsf{in})^* \wedge m \xrightarrow{\sigma} m'\}$$

F_a is the set of markings from which one can later (without the help of the environment) consume some message a. One computes F_a by backward analysis. Let G be defined by: $G = \{m \mid \exists a \in B.\ m(a) > 0 \wedge m \notin F_a\}$
G is a semi-linear set corresponding to the markings from which some message $a \in B$ will never be consumed. Then $\mathcal{N}$ is not B-consuming iff G is reachable.

B-emptying (and B-decreasing). Given an AIOPN $\mathcal{N}$ and B a subset of its channels, one decides whether $\mathcal{N}$ is B-emptying as follows. First one builds a net $\mathcal{N}'$:

- $P' = P \uplus \{run\}$
- $T' = T \uplus \{stop\}$
- $\forall p \in P\ \forall t \in T\ W'^-(p,t) = W^-(p,t), W'^+(p,t) = W^+(p,t),\ m'^0(p) = m^0(p)$
- $W'^-(run, stop) = 1,,\ W'^+(run, stop) = 0,\ m'^0(run) = 1$
- $\forall p \in P\ W'^-(p, stop) = W'^+(p, stop) = 0$
- $\forall t \in T$ such that $\lambda(t) \in \mathsf{in}\ W'^-(run, t) = W'^+(run, t) = 1$
- $\forall t \in T$ such that $\lambda(t) \notin \mathsf{in}\ W'^-(run, t) = W'^+(run, t) = 0$

$\mathcal{N}'$ behaves as $\mathcal{N}$ as long as *stop* is not fired. When *stop* is fired only transitions not labelled by inputs are fireable. Thus $\mathcal{N}$ is B-emptying iff for all $a \in B$ the set of markings $Z_a = \{m \mid m(a) = 0\}$ is a home space for $\mathcal{N}'$.

B-wholly emptying. Using the same construction $\mathcal{N}$ is B-weakly wholly emptying if $Z_B = \{m \mid m(B) = 0\}$ is a home space for $\mathcal{N}'$.

□

The next theorems, whose proofs are given in [12], establish the decidability of the necessarily properties of type (c) of Def. 10 and the communication stopping properties.

Theorem 19. *The following problems are decidable for AIOPNs: Is an AIOPN $\mathcal{N}$ necessarily B-consuming, necessarily B-decreasing, necessarily B-emptying, necessarily B-wholly emptying?*

Theorem 20. *The following problems are decidable for AIOPNs: Is an AIOPN $\mathcal{N}$ B-communication stopping, strongly B-communication stopping?*

6 Conclusion and Future Work

We have introduced asynchronously composed I/O-Petri nets and we have studied various properties of their communication channels based on a transition system semantics. Useful links between the channel properties are established. We have shown that the channel properties are compositional thus supporting incremental design. Moreover we have shown that the channel properties for AIOPNs are decidable. This work can be extended in at least three directions. The first direction would introduce new operations on AIOPNs, like hiding, to design component systems in a hierarchical way by encapsulating subsystems. The second direction concerns more general communication schemes like broadcasting. Finally, we want to establish conditions for the preservation of channel properties along the "vertical axis" namely by refinement, in particular within the framework of modal Petri nets as considered in [8].

Acknowledgement: We would like to thank the reviewers of the submitted version of this paper for many useful comments which led to a major restructuring of the paper.

References

1. Alfaro, L., Henzinger, T.: Interface-based design. In: Broy, M., Grünbauer, J., Harel, D., Hoare, T. (eds.) Engineering Theories of Software Intensive Systems, NATO Science Series, vol. 195, pp. 83–104. Springer Netherlands (2005)
2. Basu, S., Bultan, T., Ouederni, M.: Synchronizability for verification of asynchronously communicating systems. In: Kuncak, V., Rybalchenko, A. (eds.) Verification, Model Checking, and Abstract Interpretation, Lecture Notes in Computer Science, vol. 7148, pp. 56–71. Springer Berlin Heidelberg (2012)
3. Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P.: Systems and Software Verification: Model-Checking Techniques and Tools. Springer Publishing Company, Incorporated (2001)
4. Best, E., Devillers, R., Koutny, M.: Petri net algebra. Springer-Verlag New York, Inc., New York, NY, USA (2001)
5. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM 30(2), 323–342 (Apr 1983)

6. Carstensen, H.: Decidability questions for fairness in petri nets. In: Brandenburg, F., Vidal-Naquet, G., Wirsing, M. (eds.) STACS 87, Lecture Notes in Computer Science, vol. 247, pp. 396–407. Springer Berlin Heidelberg (1987)
7. Cécé, G., Finkel, A.: Verification of programs with half-duplex communication. Inf. Comput. 202(2), 166–190 (Nov 2005)
8. Elhog-Benzina, D., Haddad, S., Hennicker, R.: Refinement and asynchronous composition of modal petri nets. In: Jensen, K., Donatelli, S., Kleijn, J. (eds.) Transactions on Petri Nets and Other Models of Concurrency V, Lecture Notes in Computer Science, vol. 6900, pp. 96–120. Springer Berlin Heidelberg (2012)
9. Escrig, D.d.F., Johnen, C.: Decidability of Home Space Property. Rapports de recherche. Université Paris-Sud. Centre d'Orsay. Laboratoire de recherche en informatique, Université de Paris-Sud, Centre d'Orsay, Laboratoire de Recherche en Informatique (1989)
10. Ginsburg, S., Spanier, E.H.: Semigroups, presburger formulas and languages. Pacific Journal of Mathematics 16(2), 285–296 (1966)
11. Gomes, L., Barros, J.a.P.: Structuring and composability issues in petri nets modeling. Industrial Informatics, IEEE Transactions on 1(2), 112–123 (2005)
12. Haddad, S., Hennicker, R., Møller, M.H.: Channel properties of asynchronously composed petri nets. Tech. Rep. LSV-13-05, Laboratoire Spécification et Vérification, ENS Cachan, France (2013)
13. Hennicker, R., Janisch, S., Knapp, A.: Refinement of components in connection-safe assemblies with synchronous and asynchronous communication. In: Choppy, C., Sokolsky, O. (eds.) Foundations of Computer Software. Future Trends and Techniques for Development, Lecture Notes in Computer Science, vol. 6028, pp. 154–180. Springer Berlin Heidelberg (2010)
14. Hennicker, R., Knapp, A.: Modal interface theories for communication-safe component assemblies. In: Cerone, A., Pihlajasaari, P. (eds.) Theoretical Aspects of Computing ICTAC 2011, Lecture Notes in Computer Science, vol. 6916, pp. 135–153. Springer Berlin Heidelberg (2011)
15. Jančar, P.: Decidability of a temporal logic problem for petri nets. Theor. Comput. Sci. 74(1), 71–93 (Jul 1990)
16. Kindler, E.: A compositional partial order semantics for petri net components. In: Azéma, P., Balbo, G. (eds.) Application and Theory of Petri Nets 1997, Lecture Notes in Computer Science, vol. 1248, pp. 235–252. Springer Berlin Heidelberg (1997)
17. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: Kleijn, J., Yakovlev, A. (eds.) Petri Nets and Other Models of Concurrency - ICATPN 2007, Lecture Notes in Computer Science, vol. 4546, pp. 321–341. Springer Berlin Heidelberg (2007)
18. Mayr, E.W.: An algorithm for the general petri net reachability problem. In: Proceedings of the thirteenth annual ACM symposium on Theory of computing. pp. 238–246. STOC '81, ACM, New York, NY, USA (1981)
19. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice Hall PTR, Upper Saddle River, NJ, USA (1981)
20. Rackoff, C.: The covering and boundedness problems for vector addition systems. Theoretical Computer Science 6, 223–231 (1978)
21. Reisig, W.: Simple composition of nets. In: Franceschinis, G., Wolf, K. (eds.) Applications and Theory of Petri Nets, Lecture Notes in Computer Science, vol. 5606, pp. 23–42. Springer Berlin Heidelberg (2009)

22. Souissi, Y.: On liveness preservation by composition of nets via a set of places. In: Rozenberg, G. (ed.) Papers from the 11th International Conference on Applications and Theory of Petri Net: Advances in Petri Nets 1991, Lecture Notes in Computer Science, vol. 524, pp. 277–295. Springer Berlin Heidelberg (1991)
23. Souissi, Y., Memmi, G.: Composition of nets via a communication medium. In: Rozenberg, G. (ed.) Advances in Petri Nets 1990, Lecture Notes in Computer Science, vol. 483, pp. 457–470. Springer Berlin Heidelberg (1991)
24. Stahl, C., Vogler, W.: A trace-based service semantics guaranteeing deadlock freedom. Acta Informatica 49(2), 69–103 (2012)
25. Stahl, C., Wolf, K.: Deciding service composition and substitutability using extended operating guidelines. Data Knowl. Eng. 68(9), 819–833 (Sep 2009)
26. Valk, R., Jantzen, M.: The residue of vector sets with applications to decidability problems in petri nets. Acta Informatica 21(6), 643–674 (1985)