Process Refinement and Asynchronous Composition with Modalities

Dorsaf Elhog-Benzina[1], Serge Haddad[1], and Rolf Hennicker[2]

[1] LSV, CNRS & Ecole Normale Supérieure de Cachan
94235 CACHAN, 61 avenue du Président Wilson, France
{*elhog*, *haddad*}@lsv.ens-cachan.fr
[2] Institut für Informatik Universität München
Oettingenstraße 67 D-80538 München, Germany
hennicke@pst.ifi.lmu.de

Abstract. We propose a framework for the specification of infinite state systems based on Petri nets with distinguished *may*- and *must*-transitions (called modalities) which specify the allowed and the required behavior of refinements and hence of implementations. Formally, refinements are defined by relating the modal language specifications generated by two modal Petri nets according to the refinement relation for modal language specifications. We show that this refinement relation is decidable if the underlying modal Petri nets are weakly deterministic. We also show that the membership problem for the class of weakly deterministic modal Petri nets is decidable. As an important application of our approach we consider I/O-Petri nets which are obtained by asynchronous composition and thus exhibit inherently an infinite behavior.

Key words: Modal language specification and refinement, modal Petri net, weak determinacy, asynchronous composition, infinite state system.

1 Introduction

Specification in component-based software products. In component based software development, *specification* is an important phase of the components' life cycle. It aims to produce a formal description of the component's desired properties and behavior. A behavior specification can be presented either in terms of transition systems or in terms of logic, which both cannot be processed by a machine. Thus an *implementation phase* is required to produce concrete executable programs.

Modal specifications. Modal specifications have been introduced in [11] as a formal model for specification and implementation. A modal specification explicitly distinguishes between required actions and allowed ones. Required actions, denoted with the modality *must*, are compulsory in all possible implementations while allowed actions, denoted with the modality *may*, may happen but are not mandatory for an implementation. An implementation is seen as a specific specification in which all actions are required. Modal specifications are adequate for loose specifications as decisions can be delayed to later steps of the component's life cycle. Two different modal formalisms have been adopted in the literature, the first one, introduced in [8], is based on transition systems while the second one, introduced in [16], is a language-based model defining *modal language specifications*.

Modal refinement. A transformation step from a more abstract specification to a more concrete one is called a *refinement*. It produces a specification that is more precise, i.e. has less possible implementations. It follows that the set of implementations of a refinement is included in the set of possible implementations of the original specification.

Computational issues. While dealing with modal specifications, three main decision problems have been raised:

- (C) Consistency problem: Deciding whether a modal specification is consistent, i.e deciding whether it admits an implementation. Consistency is guaranteed for modal transition systems by definition [11] as every required transition is also an allowed one. In the case of mixed transition systems, i.e. if must-transitions are not necessarily may-transitions, the consistency decision problem is EXPTIME-complete in the size of the specification [2]. A better result is obtained with modal language specifications since a polynomial time algorithm has been proposed in [16].
- (CI) Common implementation: Deciding whether $k > 1$ modal specifications have a common implementation. Deciding common implementation of k modal transition systems is PSPACE-hard in the sum of the sizes of the k specifications [1] while it is EXPTIME-complete when dealing with mixed transition systems [2].
- (TR) Thorough refinement: Deciding whether one modal specification $\mathcal{S}$ thoroughly refines another modal specification $\mathcal{S}'$, i.e deciding whether the class of possible implementations of $\mathcal{S}$ is included in the class of possible implementations of $\mathcal{S}'$. The problem is also PSPACE-hard if $\mathcal{S}$ and $\mathcal{S}'$ are modeled with modal transition systems and it is EXPTIME-complete when they are modeled with mixed transition systems.

Limits of the existing formalisms. Both formalisms consider finite state systems. This restriction limits the existing approaches to synchronous composition. In fact asynchronous composition introduces a delay between the actions of sending and receiving a message between the communication partners. For instance, if a sender is always active while a receiver is always blocked, we naturally obtain an infinite state system.

Our contribution. We aim in this paper to deal with asynchronous composition of modal specifications while keeping most of the problems decidable. Petri nets seem to be an appropriate formalism to our needs since they allow for a finite representation of infinite state systems. Automata with queues might be another alternative for modeling infinite state systems, but all significant problems (e.g. the reachability problem) are known to be undecidable while they are decidable when considering deterministic Petri nets. In our approach we consider

Petri nets with silent transitions and we define the generalized notion of a *weakly deterministic* Petri net, as a variant of deterministic Petri nets that keeps decidability for our targeted decision problems. We also follow a language approach and use weakly deterministic Petri nets as a device to generate languages in the same way as Raclet et. al. use deterministic finite transition systems as a device to generate regular languages. Moreover, we extend Petri nets by modalities for the transitions. In this setting, we are mainly interested in the following decision problems:

1. Decide whether a given Petri net is weakly deterministic.
2. Decide whether a given language generated by a Petri net $\mathcal{N}$ is included in the language generated by a given *weakly deterministic* Petri net $\mathcal{N}'$.
3. Decide whether a given modal Petri net is (modally) weakly deterministic.
4. Given two modal language specifications $\mathcal{S}(\mathcal{M})$ and $\mathcal{S}(\mathcal{M})$ generated by two *weakly deterministic* modal Petri nets $\mathcal{M}$ and $\mathcal{M}'$ respectively, decide whether $\mathcal{S}(\mathcal{M}')$ is a modal language specification refinement of $\mathcal{S}(\mathcal{M})$.

We show that all the above mentioned problems are decidable. As a particular important application of our approach we consider I/O-Petri nets which are obtained by asynchronous composition and thus exhibit an infinite behavior. Since transitions with internal labels, in particular those obtained by internal communications, are not relevant for refinements we abstract them away by a general abstraction operator on modal I/O-Petri nets.

Outline of the paper. We proceed by reviewing in Sect. 2 modal language specifications and the associated notion of refinement. In Sect. 3 we summarize basics of Petri net theory and introduce *weakly deterministic* Petri nets. We then introduce *modal* Petri nets, their generated language specifications and we extend the concept of weak determinacy to Petri nets with modalities. In Sect. 4, we consider modal Petri nets over an I/O-alphabet (with distinguished input, output, and internal labels) and define their asynchronous composition. We also define the *abstraction* of an I/O-Petri net by relabeling internal labels to the empty word. In Sect. 5 we present the decision algorithms of the issues mentioned above. Finally we conclude in Sect. 6.

2 Modal Language Specifications

Modal specification was first introduced by Larsen and Thomsen in [11] with finite state modal transition systems by defining restrictions on specifications transitions by the mean of *may* (allowed) and *must* (required) modalities. This notion has then been adapted by Raclet in his PhD thesis who applied it to regular languages. Finite transition systems are low level models based on states, which limits the level of abstraction of the specification. They also lead to state number explosion while composing systems with an important number of states. Moreover, the complexity of the decision problems discussed above are better with modal languages than with modal transition systems. Besides, modal refinement is sound and complete with the language-based formalism while it is non-complete with transition system based formalism [9]. So a language approach is more convenient to deal with modal specification issues. Next we review the definition of modal language specification and refinement between specifications as introduced in [16].

Definition 1 (Modal language specification). *A* modal language specification $\mathcal{S}$ *over an alphabet* Σ *is a triple* $\langle \mathcal{L}, may, must \rangle$ *where* $\mathcal{L} \subseteq \Sigma^*$ *is a prefix-closed language over* Σ *and* $may, must : \mathcal{L} \to P(\Sigma)$ *are partial functions. For every trace* $u \in \mathcal{L}$*,*

- $a \in may(u)$ *means that the action* a *is allowed after* u*,*
- $a \in must(u)$ *means that the action* a *is required after* u*,*[1]
- $a \notin may(u)$ *means that* a *is forbidden after* u*.*

The modal language specification S *is* consistent *if the following two conditions hold:*
(C1) $\forall\ u \in \mathcal{L}\ must(u) \subseteq may(u)$
(C2) $\forall\ u \in \mathcal{L}\ may(u) = \{a \in \Sigma \mid u.a \in \mathcal{L}\}$

Example 1. Let us consider the example of a message producer and a message consumer represented in figure 1.

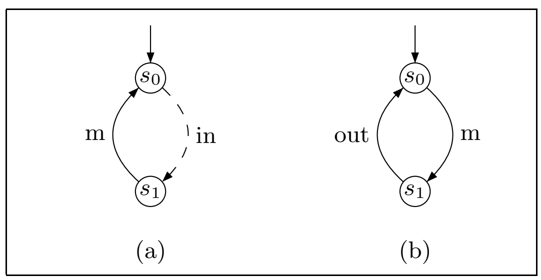

Fig. 1. Modal transition systems for a producer (a) and a consumer (b)

In the producer system, transition $s_0 \xrightarrow{in} s_1$ is allowed but not required (dashed line) while transition $s_1 \xrightarrow{m} s_0$ is required (continuous line). In the consumer model, all transitions are required. The languages associated with the the producer is $\mathcal{L} \equiv (in.m)^* + in.(m.in)^*$. The associated modal language specification is then $\langle \mathcal{L}, may, must \rangle$ with:

- $\forall\ u\ \in\ (in.m)^*\ must(u) = \emptyset \wedge may(u) = \{in\}$

[1] If $must(u)$ contains more than one element, this means that any correct implementation must have after the trace u (at least) the choice between all actions in $must(u)$.

– $\forall\ u \in in.(m.in)^*\ must(u) = may(u) = \{m\}$

Similarly, the modal language specification associated with the consumer is $\langle (m.out)^* + m.(out.m)^*, may, must\rangle$ with:

– $\forall\ u\ \in\ (m.out)^*\ must(u) = may(u) = m$
– $\forall\ u \in m.(out.m)^*\ must(u) = may(u) = out$

Modal language specifications are related by a refinement relation that translates the degree of specialization. One can obtain a possible refinement by either removing some allowed events or changing them to required events. So we review the formal definition of modal language specification refinement.

Definition 2 (Modal language specification refinement). *Let $\mathcal{S} = \langle \mathcal{L}, may, must\rangle$ and $\mathcal{S}' = \langle \mathcal{L}', may', must'\rangle$ be two consistent modal language specifications over the same alphabet Σ. $\mathcal{S}'$ is* a modal language specification refinement *of $\mathcal{S}$, denoted by $\mathcal{S}' \sqsubseteq \mathcal{S}$, if:*

– *$\mathcal{L}' \subseteq \mathcal{L}$,*
– *for every $u \in \mathcal{L}'$, $must(u) \subseteq must'(u)$, i.e every required action after the trace u in $\mathcal{L}$ is a required action after u in $\mathcal{L}'$.*

3 Modal Petri Nets

In contrast to modal language specifications, Petri nets provide an appropriate tool to specify the behavior of infinite state systems in a finitary way. Therefore we are interested in the following to combine the advantages of Petri nets with the flexibility provided by modalities for the definition of refinements. Of particular interest are Petri nets which support silent transitions and hence are able to characterize observable system behaviors. In the following we will consider such Petri nets and extend them by modalities. Then we will use modal Petri nets as a device to generate modal language specifications as the basis for refinement. First, we review basic definitions of Petri net theory and we will introduce weakly deterministic Petri nets.

3.1 Basics of Petri Nets

Definition 3 (Labeled Petri Net). *A* labeled Petri net *over an alphabet Σ is a tuple $\mathcal{N} = (P, T, W^-, W^+, \lambda, m_0)$ where:*

– *P is a finite set of places,*
– *T is a finite set of transitions with $P \cap T = \emptyset$,*
– *W^- (resp. W^+) is a matrix indexed by $P \times T$ with values in $\mathbb{N}$; W^- (resp. W^+) is called the* backward (forward) incidence matrix*,*
– *$\lambda : T \to \Sigma \cup \{\epsilon\}$ is a transition labeling function where ϵ denotes the empty word, and*
– *$m_0 : P \mapsto \mathbb{N}$ is an initial marking.*

A marking is a mapping $m : P \mapsto \mathbb{N}$. The labeling function is extended to sequences of transitions $\sigma = t_1 t_2 ... t_n \in T^*$ where $\lambda(\sigma) = \lambda(t_1)\lambda(t_2)...\lambda(t_n)$. For each $t \in T$, ${}^\bullet t$ ($t^\bullet$ resp.) denotes the set of *input (output) places* of t. i.e. ${}^\bullet t = \{p \in P \mid W^-(p,t) > 0\}$ ($t^\bullet = \{p \in P \mid W^+(p,t) > 0\}$ resp.). Likewise for each $p \in P$, ${}^\bullet p$ ($p^\bullet$) denotes the set of *input (output) transitions* of p i.e. ${}^\bullet p = \{t \in T \mid W^+(p,t) > 0\}$ ($p^\bullet = \{t \in T \mid W^-(p,t) > 0\}$ resp.). The input (output resp.) vector of a transition t is the column vector of matrix W^- (W^+ resp.) indexed by t.

In the sequel labeled Petri nets are simply called Petri nets. We have not included final markings in the definition of a Petri net here, because we are interested in potentially infinite system behaviors. We now introduce the semantic of a net.

Definition 4 (Firing rule). *Let $\mathcal{N}$ be a Petri net. A transition $t \in T$ is firable in a marking m, denoted by $m[t\rangle$, iff $\forall p \in {}^\bullet t,\ m(p) \geq W^-(p,t)$. The set of firable transitions in a marking m is defined by $firable(m) = \{t \in T \mid m[t\rangle\}$. For a marking m and $t \in firable(m)$, the firing of t from m leads to the marking m', denoted by $m[t\rangle m'$, and defined by $\forall p \in P, m'(p) = m(p) - W^-(p,t) + W^+(p,t)$.*

Definition 5 (Firing sequence). *Let $\mathcal{N}$ be a Petri net with the initial marking m_0. A finite sequence $\sigma \in T^*$ is firable in a marking m and leads to a marking m', also denoted by $m[\sigma\rangle m'$, iff either $\sigma = \epsilon$ or $\sigma = \sigma_1.t$ with $t \in T$ and there exists m_1 such that $m[\sigma_1\rangle m_1$ and $m_1[t\rangle m'$. For a marking m and $\sigma \in T^*$, we write $m[\sigma\rangle$ if σ is firable in m. The set of reachable markings is defined by $Reach(\mathcal{N}, m_0) = \{m \mid \exists \sigma \in T^* \text{such that } m_0[\sigma\rangle m\}$.*

The reachable markings of a Petri net correspond to the reachable states of the modeled system. Since the capacity of places are not restricted, the set of the reachable markings of the Petri nets considered here may be infinite. Thus, Petri nets can model infinite state systems. Now, let us define the language generated by a labeled Petri net.

Definition 6 (Petri net language). *Let $\mathcal{N}$ be a labeled Petri net over the alphabet Σ. The language generated by $\mathcal{N}$ is:*

$$\mathcal{L}(\mathcal{N}) = \{u \in \Sigma^* \mid \exists \sigma \in T^* \text{ and } m \text{ such that } \lambda(\sigma) = u \text{ and } m_0[\sigma\rangle m\} \ .$$

A particular interesting class of Petri nets are deterministic Petri nets as defined, e.g., in [14]. Since in our approach we also deal with silent transitions, which are not taken into account for the generated languages, we are interested in a more general notion of determinacy which allows us to abstract from silent transitions. This leads to our notion of weakly deterministic Petri net. We call a Petri net weakly deterministic if any two firing sequences σ and σ' which produce the same word u can be mutually extended to produce the same continuations of u. In this sense our notion of weak deterministic Petri net corresponds to Milner's (weak) determinacy [13] and to the concept of a weakly deterministic transition system in [5]. It is also related to to the notion of output-determinacy in [7].

Definition 7 (Weakly Deterministic Petri net). *Let $\mathcal{N}$ be a labeled Petri net with initial marking m_0 and labeling function $\lambda : T \to \Sigma \cup \{\epsilon\}$. For any marking m, let*

$$may_{mk}(m) = \{a \in \Sigma \mid \exists \sigma \in T^* \text{ such that } \lambda(\sigma) = a \text{ and } m[\sigma\rangle\}$$

$\mathcal{N}$ is called weakly deterministic*, if for each $\sigma, \sigma' \in T^*$ with $\lambda(\sigma) = \lambda(\sigma')$ and for any markings m and m' with $m_0[\sigma\rangle m$ and $m_0[\sigma'\rangle m'$, we have $may_{mk}(m) = may_{mk}(m')$.*

Since weakly deterministic Petri nets play an important role in our further development it is crucial to know, whether a given Petri net belongs to the class of weakly deterministic Petri nets. This leads to our first decision problem stated below. Our second decision problem is motivated by the major goal of this work to provide formal support for refinement in system development. Since, in a simple form, refinement can be defined by language inclusion, we want to be able to decide this. Unfortunately, it is well-known that the language inclusion problem for Petri nets is undecidable [4]. However, in [14] it has been shown that for languages generated by deterministic Petri nets the language inclusion problem is decidable. Therefore we are interested in a generalization of this result for languages generated by weakly deterministic Petri nets which leads to our second decision problem. Observe that we do not require $\mathcal{N}'$ to be weakly deterministic.

First decision problem. Given a labeled Petri net $\mathcal{N}$, decide whether $\mathcal{N}$ is weakly deterministic.

Second decision problem. Let $\mathcal{L}(\mathcal{N})$ and $\mathcal{L}(\mathcal{N}')$ be two languages with the same alphabet Σ such that $\mathcal{L}(\mathcal{N})$ is generated by a weakly deterministic Petri net $\mathcal{N}$ and $\mathcal{L}(\mathcal{N}')$ is generated by a Petri net $\mathcal{N}'$. Decide whether $\mathcal{L}(\mathcal{N}')$ is included in $\mathcal{L}(\mathcal{N})$.

3.2 Modal Petri Nets

In the following we introduce modal Petri nets which extend, similarly to modal language specifications, Petri nets with modalities *may* and *must* on its transitions.

Definition 8 (Modal Petri net). *A* modal Petri net *$\mathcal{M}$ over an alphabet Σ is a pair $\mathcal{M} = (\mathcal{N}, T_\square)$ where $\mathcal{N} = (P, T, W^-, W^+, \lambda, m_0)$ is a labeled Petri net over Σ and $T_\square \subseteq T$ is a set of* must (required) *transitions. The set of* may (allowed) *transitions is the set of transitions T.*

Example 2. Let us consider the same example of a message producer and a message consumer (see Fig. 2). The consumer may receive an input *in* (white transition) but must produce a message *m* (black transition). The consumer must receive a message *m* and produce an output *out*.

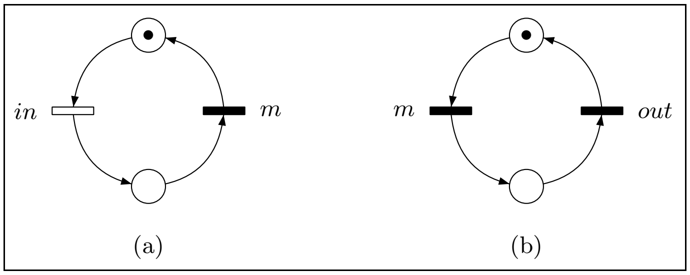

Fig. 2. Modal Petri nets for a producer (a) and a consumer (b)

Any modal Petri net $\mathcal{M} = (\mathcal{N}, T_\Box)$ gives rise to the construction of a modal language specification (see Def. 1) which extends the language $\mathcal{L}(\mathcal{N})$ by *may* and *must* modalities. Similarly to the construction of $\mathcal{L}(\mathcal{N})$ the definition of the modalities should abstract from silent transitions in an appropriate way. While for the *may* modality this is rather straightforward, special care has to be taken for the definition of the *must* modality. For this purpose, we introduce the following auxiliary definition which expresses, for each marking m, the set $must_{mk}(m)$ of all labels $a \in \Sigma$ which must be produced by firing (in m) some silent *must*-transitions succeeded by a *must*-transition labeled by a. This means that the label a must be produced as the next visible label by some firing sequence of m. Formally, for any marking m, let

$$must_{mk}(m) = \{a \in \Sigma \mid \exists \sigma \in T_\Box^*, t \in T_\Box \text{ such that } \lambda(\sigma) = \epsilon, \lambda(t) = a \text{ and } m[\sigma t\rangle\}$$

On this basis we can now compute for each word $u \in \mathcal{L}(\mathcal{N})$ and for each marking m reachable by firing a sequence of transitions which produces u (and which has no silent transition at the end[2]), the set $must_{mk}(m)$. Then the labels in $must_{mk}(m)$ must be the possible continuations of u in the generated modal language specification.

Definition 9 (Modal Petri Net Language Specification). *Let $\mathcal{M} = (\mathcal{N}, T_\Box)$ be a modal Petri net over an alphabet Σ such that $\lambda : T \to \Sigma \cup \{\epsilon\}$ is the labeling function and m_0 is the initial marking of $\mathcal{N}$. $\mathcal{M}$ generates the* modal language specification *$\mathcal{S}(\mathcal{M}) = \langle \mathcal{L}(\mathcal{N}), may, must\rangle$ where:*

- *$\mathcal{L}(\mathcal{N})$ is the language generated by the Petri net $\mathcal{N}$,*
- *$\forall u \in \mathcal{L}(\mathcal{N})$, $may(u) =$ $\{a \in \Sigma \mid \exists \sigma \in T^* \text{ and } m \text{ such that } \lambda(\sigma) = u, m_0[\sigma\rangle m \text{ and } a \in may_{mk}(m)\}$,*
- *$\forall u \in \mathcal{L}(\mathcal{N}), x \in \Sigma$,*
  - *$must(\epsilon) = must_{mk}(m_0)$,*
  - *$must(ux) = \{a \in \Sigma \mid \exists \sigma \in T^*, t \in T \text{ and } m \text{ such that } \lambda(\sigma) = u, \lambda(t) = x, m_0[\sigma t\rangle m \text{ and } a \in must_{mk}(m)\}$.*

[2] We require it to avoid false detection of must transitions starting from intermediate markings.

Remark 1. Any modal language specification generated by a modal Petri net is consistent.

Example 3. Let us consider the modal Petri net in Fig. 3.

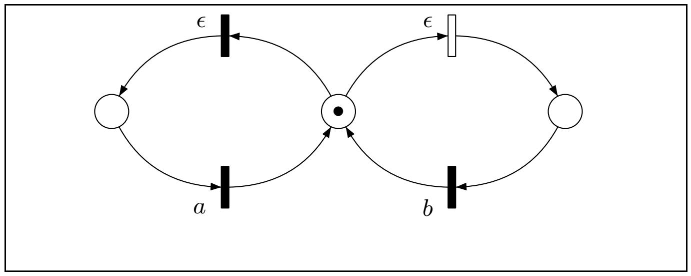

Fig. 3. Modal Petri net with silent transitions

The modal language specification generated by this net consists of the language $\mathcal{L}$ presented by the regular expression $(a^*b^*)^*$ and of the modalities $may(u) = \{a, b\}$, and $must(u) = \{a\}$ for $u \in \mathcal{L}$. Note that b is not a *must* as it is preceeded by a silent *may*-transition (which can be omitted in a refinement).

The notion of weakly deterministic Petri net can be extended to modal Petri nets by taking into account an additional condition for *must*-transitions. This condition ensures that for any two firing sequences σ and σ' which produce the same word u, the continuations of u produced by firing sequences of *must*-transitions after σ and σ' are the same.

Definition 10 (Weakly Deterministic Modal Petri Net). *Let $\mathcal{M} = (\mathcal{N}, T_\Box)$ be a modal Petri net over an alphabet Σ such that $\lambda : T \to \Sigma \cup \{\epsilon\}$ is the labeling function of $\mathcal{N}$. For any marking m, let*

$$must_{mk}(m) = \{a \in \Sigma \mid \exists \sigma \in T_\Box^*, t \in T_\Box \text{ such that } \lambda(\sigma) = \epsilon, \lambda(t) = a \text{ and } m[\sigma t\rangle\}$$

$\mathcal{M}$ is (modally) weakly deterministic*, if*

1. *$\mathcal{N}$ is weakly deterministic, and*
2. *for each $\sigma, \sigma' \in T^*$ with $\lambda(\sigma) = \lambda(\sigma')$ and for any markings m and m' with $m_0[\sigma\rangle m$ and $m_0[\sigma'\rangle m'$, we have $must_{mk}(m) = must_{mk}(m')$.*

Remark 2. For any weakly deterministic modal Petri net $\mathcal{M} = (\mathcal{N}, T_\Box)$ the definition of the modalities of its generated modal language specification $\mathcal{L}(\mathcal{M}) = \langle \mathcal{L}(\mathcal{N}), may, must \rangle$ can be simplified as follows:

– $\forall u \in \mathcal{L}(\mathcal{N})$, let $\sigma \in T^*$ and let m be a marking such that $\lambda(\sigma) = u$ and $m_0[\sigma\rangle m$, then $may(u) = may_{mk}(m)$.

– $\forall u \in \mathcal{L}(\mathcal{N}), x \in \Sigma$, let $\sigma \in T^*, t \in T$ and let m be a marking such that $\lambda(\sigma) = u, \lambda(t) = x$ and $m_0[\sigma t\rangle m$, then $must(ux) = must_{mk}(m)$. Moreover, $must(\epsilon) = must_{mk}(m_0)$.

Example 4. Let us consider the modal Petri net in Fig. 4.

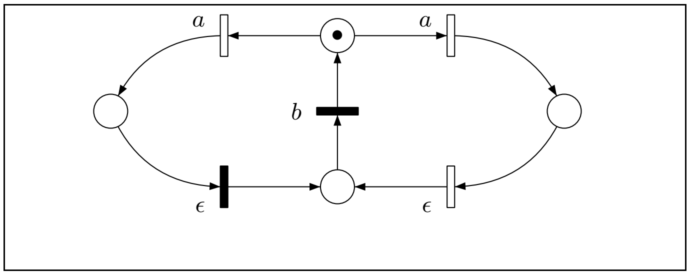

Fig. 4. Non weakly deterministic modal Petri net

Let t_l (t_r resp.) be the left (right resp.) transition labeled with a and let m_l (m_r resp.) be the marking obtained by firing the transition t_l (t_r resp.). Obviously, both transitions produce the same letter but $must(m_l) = \{b\}$ while $must(m_r) = \emptyset$ (since the silent transition firable in m_r is only a *may*-transition).

The two decision problems of Sect. 3.1 induce the following obvious extensions in the context of modal Petri nets and their generated modal language specifications. Observe that for the refinement problem we require that both nets are weakly deterministic.

Third decision problem. Given a modal Petri net $\mathcal{M}$, decide whether $\mathcal{M}$ is (modally) weakly deterministic.

Fourth decision problem. Let $\mathcal{S}(\mathcal{M})$ and $\mathcal{S}(\mathcal{M}')$ be two modal language specifications over the same alphabet Σ such that $\mathcal{S}(\mathcal{M})$ ($\mathcal{S}(\mathcal{M}')$ resp.) is generated by a weakly deterministic modal Petri net $\mathcal{M}$ ($\mathcal{M}$ resp.). Decide whether $\mathcal{S}(\mathcal{M}')$ is a modal language specification refinement of $\mathcal{S}(\mathcal{M})$.

4 Modal I/O-Petri Nets

In this section we consider modal Petri nets where the underlying alphabet Σ is partitioned into disjoint sets *in*, *out*, and *int* of input, output and internal labels resp., i.e $\Sigma = in \uplus out \uplus int$. Such alphabets are called *I/O-alphabets* and modal Petri nets over an I/O-alphabet are called *modal I/O-Petri nets*. The discrimination of input, output and internal labels provides a means to specify the communication abilities of a Petri net and hence provides an appropriate basis for Petri net composition. A syntactic requirement for the composability of two modal I/O-Petri nets is that their labels overlap only on complementary types, i.e. their underlying alphabets must be composable. Formally, two I/O-alphabets $\Sigma_1 = in_1 \uplus out_1 \uplus int_1$ and $\Sigma_2 = in_2 \uplus out_2 \uplus int_2$ are *composable* if $\Sigma_1 \cap \Sigma_2 \subseteq (in_1 \cap out_2) \cup (in_2 \cap out_1)$.[3]

Definition 11 (Alphabet Composition). *Let $\Sigma_1 = in_1 \uplus out_1 \uplus int_1$ and $\Sigma_2 = in_2 \uplus out_2 \uplus int_2$ be two composable I/O-alphabets. The composition of Σ_1 and Σ_2 is the I/O-alphabet $\Sigma_c = in_c \uplus out_c \uplus int_c$ where:*

- $in_c = (in_1 \setminus out_2) \uplus (in_2 \setminus out_1)$,
- $out_c = (out_1 \setminus in_2) \uplus (out_2 \setminus in_1)$,
- $int_c = \{*a \mid * \in \{!, ?\},\ a \in \Sigma_1 \cap \Sigma_2\} \uplus int_1 \uplus int_2$.

The input and output labels of the alphabet composition are the input and output labels of the underlying alphabets which are not used for communication, and hence are "left open". The internal labels of the alphabet composition are obtained from the internal labels of the underlying alphabets and from their shared input/output labels. Since we are interested here in asynchronous communication each shared label a is duplicated to $!a$ and $?a$ where the former represents the asynchronous sending of a message and the latter represents the receiption of the message (at some later point in time). We are now able to define the asynchronous composition of composable modal I/O-Petri nets. For the realization of the asynchronous communication, for each shared label a a new place p_a is introduced in the composition.

Definition 12 (Asynchronous Composition). *Let $\mathcal{M}_1 = (\mathcal{N}_1, T_{1_\Box})$, $\mathcal{N}_1 = (P_1, T_1, W_1^-, W_1^+, \lambda_1, m_{1_0})$ be a modal I/O-Petri net over the I/O-alphabet $\Sigma_1 = in_1 \uplus out_1 \uplus int_1$ and let $\mathcal{M}_2 = (\mathcal{N}_2, T_{2_\Box})$, $\mathcal{N}_2 = (P_2, T_2, W_2^-, W_2^+, \lambda_2, m_{2_0})$ be a modal I/O-Petri net over the I/O-alphabet $\Sigma_2 = in_2 \uplus out_2 \uplus int_2$. $\mathcal{M}_1$ and $\mathcal{M}_2$ are* composable *if $P_1 \cap P_2 = \emptyset$, $T_1 \cap T_2 = \emptyset$ and if Σ_1 and Σ_2 are composable. In this case, their* asynchronous composition *$\mathcal{M}_c$, also denoted by $\mathcal{M}_1 \otimes_{as} \mathcal{M}_2$, is the modal Petri net over the alphabet composition Σ_c, defined as follows:*

- $P_c = P_1 \uplus P_2 \uplus \{p_a \mid a \in \Sigma_1 \cap \Sigma_2\}$ *(each p_a is a new place)*
- $T_c = T_1 \uplus T_2$ *and* $T_{c,\Box} = T_{1_\Box} \uplus T_{2_\Box}$
- W_c^- *(resp. W_c^+) is the $P_c \times T_c$ backward (forward) incidence matrix defined by:*
  - *for each $p \in P_1 \cup P_2,\ t \in T_c$,*

$$W_c^-(p,t) = \begin{cases} W_1^-(p,t) & \textit{if } p \in P_1 \textit{ and } t \in T_1 \\ W_2^-(p,t) & \textit{if } p \in P_2 \textit{ and } t \in T_2 \\ 0 & \textit{otherwise} \end{cases}$$

[3] Note that for composable alphabets $in_1 \cap in_2 = \emptyset$ and $out_1 \cap out_2 = \emptyset$.

$$W_c^+(p,t) = \begin{cases} W_1^+(p,t) \; \textit{if } p \in P_1 \textit{ and } t \in T_1 \\ W_2^+(p,t) \; \textit{if } p \in P_2 \textit{ and } t \in T_2 \\ 0 \qquad\qquad \textit{otherwise} \end{cases}$$

- *for each* $p_a \in P_c \setminus \{P_1 \cup P_2\}$ *with* $a \in \Sigma_1 \cap \Sigma_2$ *and for each* $t \in T_i$ *with* $i \in \{1,2\}$,

$$W_c^-(p_a,t) = \begin{cases} 1 \textit{ if } a = \lambda_i(t) \in in_i \cap out_j \textit{ with } i \neq j \\ 0 \textit{ otherwise} \end{cases}$$

$$W_c^+(p_a,t) = \begin{cases} 1 \textit{ if } a = \lambda_i(t) \in in_j \cap out_i \textit{ with } i \neq j \\ 0 \textit{ otherwise} \end{cases}$$

– $\lambda_c : T_c \to \Sigma_c$ *is defined, for all* $t \in T_c$ *and for* $i \in \{1,2\}$, *by*

$$\lambda_c(t) = \begin{cases} \lambda_i(t) \quad \textit{if } t \in T_i,\ \lambda_i(t) \notin \Sigma_1 \cap \Sigma_2 \\ ?\lambda_i(t) \textit{ if } t \in T_i,\ \lambda_i(t) \in in_i \cap out_j \textit{ with } i \neq j \\ !\lambda_i(t) \textit{ if } t \in T_i,\ \lambda_i(t) \in in_j \cap out_i \textit{ with } i \neq j \end{cases}$$

– m_{c_0} *is defined, for each place* $p \in P_c$, *by*

$$m_{c_0}(p) = \begin{cases} m_{1_0}(p) \textit{ if } p \in P_1 \\ m_{2_0}(p) \textit{ if } p \in P_2 \\ 0 \qquad\quad \textit{otherwise} \end{cases}$$

Proposition 1. *The asynchronous composition of two weakly deterministic modal I/O-Petri nets is again a weakly deterministic modal I/O-Petri net.*

We will not give a proof of this fact here, since it is not a main result of this work.

Example 5. We consider the two modal producer and consumer Petri nets of Fig. 2 as I/O-nets where the producer alphabet has the input label *in*, the output label m and no internal labels while the consumer has the input label m, the output label *out* and no internal labels as well. Obviously, both nets are composable and their asynchronous composition yields the net shown in Fig. 5. The alphabet of the composed net has the input label *in*, the output label *out* and the internal labels $?m$ and $!m$. The Petri net composition describes an infinite state system and its generated modal language specification has a language which is no more regular.

When studying refinements it is crucial to rely on the observable behavior specified by a requirements specification while one can abstract from internal actions performed by a more concrete specification or an implementation. An important case is the situation where the concrete specification is given by the composition of (already available) specifications of single components. Then their

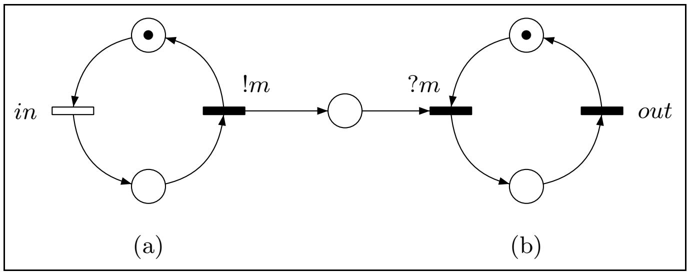

Fig. 5. Composition of the producer and consumer Petri nets

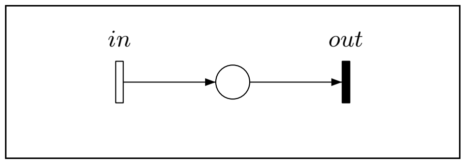

Fig. 6. Requirements specification for an infinite state producer/consumer system

composition must exhibit the required observable behaviour of a given abstract specification.

As an example we consider the requirements specification for an infinite state producer/consumer system presented by the modal I/O-Petri net in Fig. 6. Obviously, the asynchronous composition of the single producer and consumer nets in Fig. 5 is not (yet) a correct refinement since there are still the internal labels which do not correspond to silent transitions (yet) and therefore must be taken into account when comparing the two generated modal language specifications. However, since internal lables describe internal actions which are invisible from the outside, we can apply an abstraction to the composition which relabels all internal actions to the empty word ϵ. In the example the internal label are just the labels $!m$ and $?m$ used for the communication but, in general, transitions with internal labels may also describe internal computation steps of an implementation and then its is also meaningful to abstract them away. Hence we define a general abstraction operator which can be applied to any (modal) I/O-Petri net.

Definition 13 (Abstraction). *Let $\mathcal{M} = (\mathcal{N}, T_\Box)$ be a modal I/O-Petri net over the I/O-alphabet $\Sigma = in \uplus out \uplus int$ with underlying Petri net $\mathcal{N} = (P, T, W^-, W^+, \lambda, m_0)$. Let $\alpha(\Sigma) = in \uplus out \uplus \emptyset$ and let $\alpha : \Sigma \cup \{\epsilon\} \to \alpha(\Sigma) \cup \{\epsilon\}$ be the relabeling defined by $\alpha(a) = a$ if $a \in in \uplus out$, $\alpha(a) = \epsilon$ otherwise. Then the* abstraction *from $\mathcal{M}$ is the modal I/O-Petri net $\alpha(\mathcal{M}) = (\alpha(\mathcal{N}), T_\Box)$ over the I/O-alphabet $\alpha(\Sigma)$ with underlying Petri net $\alpha(\mathcal{N}) = (P, T, W^-, W^+, \alpha \circ \lambda, m_0)$.*

Coming back to our example the “abstract” Petri net in Fig. 6 is obviously modally weakly deterministic. For the abstraction of the composed Petri nets in Fig. 5 this is not obvious but, according to the results of the next section, we can decide it (and get a positive answer). Note, however, that in the case where one of the transitions used for the communication would not be a “must” the abstraction of the Petri net composition would not satisy the second condition of modal weak determinacy. The next problem is to decide whether the refinement relation holds between their generated modal language specifications. Again, according to the results of the next section, we can decide this (and also get a positive answer).

5 Decision Algorithms

We begin this section by some recalls about semi-linear sets and decision procedures in Petri nets.

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* 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 by 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 [12]. Furthermore this procedure can be adapted to semi-linear sets. Given a semi-linear set E of markings, in order to decide whether there exists a marking of E which is reachable, we proceed as follows. For any linear set $E' = \{v_0 + \sum_{1 \leq i \leq n} \lambda_i v_i \mid \forall i \ \lambda_i \in \mathbb{N}\}$ associated with E we build a net $\mathcal{N}_{E'}$ by adding transitions $t_1, \ldots, t_n$. Transition t_i has v_i as input vector and the null vector as output vector. Then one checks whether v_0 is reachable in $\mathcal{N}_{E'}$. E is reachable from m_0 iff one of these tests is positive.

In [18] 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 a transition t, the set of markings m from which there exists a transition sequence σ with $m[\sigma t\rangle$ is effectively computable.

Now we solve the decision problems stated in the previous sections.

Proposition 2. *Let $\mathcal{N}$ be a labeled Petri net, then it is decidable whether $\mathcal{N}$ is weakly deterministic.*

Proof. First we build a net $\mathcal{N}'$ defined as follows.

- Its set of places is the union of two disjoint copies P_1 and P_2 of P.
- There is one transition (t, t') for every t and t' s.t. $\lambda(t) = \lambda(t') \neq \varepsilon$. The input (resp. output) vector of this transition is the one of t with P substituted by P_1 plus the one of t' with P substituted by P_2.
- There are two transitions t_1, t_2 for every t s.t. $\lambda(t) = \varepsilon$. The input (resp. output) vector of t_1 (resp t_2) is the one of t with P substituted by P_1 (resp. P_2).
- The initial marking is m_0 with P substituted by P_1 plus m_0 with P substituted by P_2.

Then for every $a \in \Sigma$, we compute a representation of the set E_a from which, in $\mathcal{N}$ a transition labelled by a is eventually fireable after the firing of silent transitions (using results of [18]) and a representation of its complementary set $\overline{E_a}$. Afterwards we compute the representation of the semi-linear set F_a whose projection on P_1 is a vector of E_a with P substituted by P_1 and whose projection on P_2 is a vector of $\overline{E_a}$ with P substituted by P_2. Let $F = \bigcup_{a \in \Sigma} F_a$ then $\mathcal{N}$ is weakly deterministic iff F is not reachable which is decidable.

Proposition 3. *Let $\mathcal{N}$ be a weakly deterministic labeled Petri net and $\mathcal{N}'$ be a labeled Petri net then it is decidable whether $\mathcal{L}(\mathcal{N}) \subseteq \mathcal{L}'(\mathcal{N}')$.*

Proof. W.l.o.g. we assume that P and P' are disjoint. First we build a net $\mathcal{N}''$ defined as follows.

- Its set of places is the union of P and P'.
- There is one transition (t, t') for every $t \in T$ and $t' \in T'$ s.t. $\lambda(t) = \lambda(t') \neq \varepsilon$. The input (resp. output) vector of this transition is the one of t plus the one of t'.
- Every transition $t \in T \cup T'$ s.t. $\lambda(t) = \varepsilon$ is a transtion of $\mathcal{N}''$.
- The initial marking is the $m_0 + m'_0$.

Then for every $a \in \Sigma$, we compute a representation of the set $E_{\mathcal{N},a}$ (resp. $E_{\mathcal{N}',a}$) from which in $\mathcal{N}$ a transition labelled by a is eventually fireable preceeded only by silent transitions and a representation of its complementary set $\overline{E_{\mathcal{N},a}}$ (resp. $\overline{E_{\mathcal{N}',a}}$). Afterwards we compute the representation of the semi-linear set F_a whose projection on P is a vector of $\overline{E_{\mathcal{N},a}}$ and whose projection on P' is a vector of $E_{\mathcal{N}',a}$. Let $F = \bigcup_{a \in \Sigma} F_a$ then $\mathcal{L}(\mathcal{N}) \subseteq \mathcal{L}'(\mathcal{N}')$ iff F is not reachable. This procedure is sound. Indeed assume that some marking $(m, m') \in F_a$ is reachable in $\mathcal{N}''$ witnessing that after some word w, some firing sequences $\sigma \in \mathcal{N}, \sigma' \in \mathcal{N}'$ s.t. $m_0[\sigma\rangle m$, $m'_0[\sigma'\rangle m'$ and $\lambda(\sigma) = \lambda'(\sigma')$ from m one cannot "observe" a and from m' one can "observe" a. Then due to weak determinism of $\mathcal{N}$ for every m^* s.t. there exists a sequence σ^* with $m_0[\sigma^*\rangle m^*$ and $\lambda(\sigma^*) = \lambda(\sigma)$, m^* is also in $\overline{E_{\mathcal{N},a}}$.

Proposition 4. *Let $\mathcal{M}$ be a modal Petri net, then it is decidable whether $\mathcal{M}$ is (modally) weakly deterministic.*

Proof. Observe that the first condition for being weakly deterministic is decidable by proposition 2. In order to decide the second condition, we build as in the corresponding proof the net $\mathcal{N}'$. Then we build representations for the following semi-linear sets. G_a is the set of markings m of $\mathcal{N}$ such that from m a transition of $T_\Box$ labelled by a is eventually fireable after firing silent transitions of $T_\Box$. Afterwards we compute the representation of the semi-linear set H_a whose projection on P_1 is a vector of G_a with P substituted by P_1 and whose projection on P_2 is a vector of $\overline{G_a}$ with P substituted by P_2. Let $H = \bigcup_{a \in \Sigma} H_a$ then $\mathcal{M}$ fulfills the second condition of weak determinism iff H is not reachable.

Proposition 5. *Let $\mathcal{M}, \mathcal{M}'$ be two weakly deterministic modal Petri nets then it is decidable whether the modal specification $\mathcal{S}(\mathcal{M})$ refines $\mathcal{S}(\mathcal{M}')$.*

Proof. Observe that the first condition for refinement is decidable by proposition 3. In order to decide the second condition, we build as in the corresponding proof the net $\mathcal{N}''$. Then we build representations for the semi-linear sets G_a (as in the previous proof) and similarly G'_a in the case of $\mathcal{N}'$. Afterwards we compute the representation of the semi-linear set H_a whose projection on P is a vector of G_a and whose projection on P' is a vector of $\overline{G'_a}$. Let $H = \bigcup_{a \in \Sigma} H_a$ then the second condition for refinement holds iff H is not reachable. This procedure is sound. Indeed assume that some marking $(m, m') \in H_a$ is reachable in $\mathcal{N}''$ witnessing that after some word w, some firing sequences $\sigma \in \mathcal{N}, \sigma' \in \mathcal{N}'$ s.t. $m_0[\sigma\rangle m$, $m'_0[\sigma'\rangle m'$ and $\lambda(\sigma) = \lambda'(\sigma') = w$ and from m one can "observe" b by a "must" sequence and from m' one cannot observe a by a must sequence. Then due to (the second condition of) weak determinism of $\mathcal{N}'$ for every m^* s.t. there exists a sequence σ^* with $m'_0[\sigma^*\rangle m^*$ and $\lambda'(\sigma^*) = \lambda'(\sigma')$, m^* is also in $\overline{G'_a}$.

6 Conclusion

In the present work, we have introduced modal I/O-Petri nets and we have provided decision procedures to decide whether such Petri nets are weakly deterministic and whether two modal language specifications generated by weakly deterministic modal Petri nets are related by the modal refinement relation. An important role has been played by the notion of modal weak determinacy and by the abstraction operator which considers internal transitions to be silent. Since, in general, the abstraction operator does not preserve modal weak determinacy, we are interested in the investigation of conditions which ensure this preservation property. This concerns also conditions for single components such that the abstraction of their composition is modally weakly deterministic. Another direction of future research concerns the study of compatibility of component behaviours represented by modal I/O-Petri nets and the establishment of an interface theory for this framework along the lines of [3].

References

1. A. Antonik, M. Huth, K. G. Larsen, U. Nyman and A. Wasowski. Complexity of decision problems for mixed and modal specifications. In *Proc. of the 10th Int.*

Conf. on Found. of Software Science and Comp. Struct. (FoSSaCS'08), vol. 4962 of *LNCS*, Springer, 2008.
2. A. Antonik, M. Huth, K.G. Larsen, U. Nyman and A. Wasowski. EXPTIME-complete decision problems for mixed and modal specifications. In: *Proc. of EXPRESS*, July 2008.
3. S. Bauer and P. Mayer and A. Schroeder and R. Hennicker. On Weak Modal Compatibility, Refinement, and the MIO Workbench. In Proc. 16^{th} Int. Conf. Tools and Algor. for the Constr. and Analysis of Systems (TACAS'10), 2010
4. M.H.T. Hack. Decidability questions for Petri Nets. Ph.D.Thesis. M.I.T (1976)
5. R. Hennicker and S. Janisch and A. Knapp. On the Observable Behaviour of Composite Components. In Electr. Notes Theor. Comput. Sci. Volume 260, 2010, pages 125-153. http://dx.doi.org/10.1016/j.entcs.2009.12.035, DBLP, http://dblp.uni-trier.de
6. R.M. Karp, R.E. Miller Parallel program schemata. In: *JTSS* 4, 1969, pp 147-195.
7. V. Khomenko, M. Schaefer and W. Vogler. Output-Determinacy and Asynchronous Circuit Synthesis. In: Fundam. Inf. 88, 4, pages 541-579 (Dec. 2008)
8. K.G. Larsen. Modal specifications. In: *Joseph Sifakis, editor, Automatic Verification Methods for Finite State Systems*, volume 407 of LNCS, pages 232–246, 1989.
9. K.G. Larsen, U. Nyman and A. Wasowski. On modal refinement and consistency. In *Proc. of the 18th International Conference on Concurrency Theory, (CONCUR07)*, LNCS pages 105–119, Springer Verlag, 2007.
10. K.G. Larsen, U. Nyman and A. Wasowski. Modal I/O automata for interface and product line theories. In *Programming Languages and Systems, 16th European Symposium on Programming (ESOP07)*, vol. 4421 of *LNCS*, pages 64–79. Springer, 2007.
11. K.G. Larsen and b. Thomsen. A modal process logic. In: *Third Annual IEEE Symposium on Logic in Computer Science LICS*, pages 203–210, 1988.
12. E. Mayr. An algorithm for the general Petri net reachability problem. In *Proc. of the 13th Annual ACM Symposium on Theory of Computing (STOC'81)* pages 238–246, 1981.
13. R. Milner Communication and concurrency. Prentice Hall, 1989.
14. E. Pelz. Closure properties of deterministic Petri net languages. In *STACS'87*, vol. 247 of *LNCS*, Springer 1987.
15. J.L. Peterson, Petri net theory and the modeling of systems, Prentice-Hall, Englewood Cliffs, NJ, 1981
16. J.-B. Raclet. Residual for Component Specifications. In: *Proc. of the 4th International Workshop on Formal Aspects of Component Software (FACS07), Sophia-Antipolis, France*, September 2007.
17. J.-B. Raclet, E. Badouel, A. Benveniste, B. Caillaud, A. Legay and R. Passerone. Modal Interfaces: Unifying Interface Automata and Modal Specifications. In *Proc. of 9th International Conference on Embedded Software (EMSOFT'09), Grenoble, France, ACM*, October 2009.
18. R. Valk, M. Jantzen The residue of vector sets with applications to decidability problems in Petri nets Advances in Petri Nets 1984, LNCS volume 188 pp. 234-258