Refinement and Asynchronous Composition of Modal Petri Nets

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
hennicker@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. For any modal Petri net, we define its generated modal language specification which abstracts away silent transitions. On this basis we consider refinements of modal Petri nets by relating their generated 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 scenario of our approach we consider I/O-Petri nets and their asynchronous composition which typically leads to an infinite state system.

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

1 Introduction

In component-based software development, *specification* is an important phase in the life cycle of each component. 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 directly executed by a machine. Thus an *implementation phase* is required to produce concrete executable programs.

Modal specifications have been introduced in [18] as a formal model for specification and implementation. A modal specification explicitly distinguishes between required transitions and allowed ones. Required transitions, denoted with the modality *must*, are compulsory in all correct implementations while allowed transitions, denoted with the modality *may*, may occur but can also be omitted in an implementation. An implementation is seen as a particular specification in which all transitions are required. Thus modalities support underspecification as well as tight specifications saying that certain activities must be present.

Therefore they provide a flexible tool for system development 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 [15], is based on transition systems while the second one, introduced in [24], is a language-based model defining *modal language specifications*.

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. Hence the set of implementations of a refinement is included in the set of possible implementations of the original specification. From the practical and from the computational point of view, it is of course an important issue to be able to check refinements, and even better, to decide whether a refinement relation holds. For modal transition systems with finite states and for modal language specifications whose underlying languages are regular, the refinement problem is decidable.

Due to its simplicity and appropriateness to design, this modal approach has led both to industrial case studies, see, e.g., [17], and to theoretical developments in order to integrate timing requirements [6] or probabilistic requirements [8]. However, none of these works deals with infinite state systems or asynchronous composition while in distributed systems such features are indispensable. For instance, (discrete) infinite state systems can easily appear by the asynchronous composition of components, since asynchronous communication introduces a delay between the actions of sending and receiving a message between the communication partners. Indeed the size of every communication channel is potentially unbounded. But also requirement specifications for complex systems may involve infinite state spaces. Thus the motivation of our work was to extend the modal approach to take into account infinite state system specifications and their modal refinement while keeping most of the problems decidable. In particular, our results should be applicable to systems of asynchronoulsy communicating components.

Our contribution. Petri nets are an appropriate formalism for our needs since they allow for a finite representation of infinite state systems. Automata with queues might be another alternative, but all significant problems (e.g. the reachability problem) are known to be undecidable [7] while they are decidable when considering deterministic Petri nets [21]. In our approach, we consider Petri nets with silent transitions labeled by ϵ. Silent transitions are invisible to the environment and hence are the basis for observational abstraction. This is particularly important to obtain a powerful refinement relation which relies only on observable behaviors. In our approach we define the generated language of a Petri net by abstracting away silent transitions. Then we consider Petri net refinement as inclusion of the generated languages which means that the observable execution traces of a "concrete" (refining) Petri net must be observable execution traces of the "abstract" (refined) Petri net as well. We know from [21] that for languages generated by deterministic Petri nets the language inclusion problem is decidable but also, from [10], that in general the language inclusion problem for Petri nets is undecidable. In the presence of silent transitions we are, unfortunately, very

often in a situation where Petri nets have non-deterministic silent choices such that we cannot use the decidability results for the deterministic case. Therefore, we introduce the generalized notion of *weakly deterministic* Petri nets, which are deterministic "up to silent transitions". We show that the following problems are decidable:

1. Decide whether a given Petri net is weakly deterministic.
2. Decide whether a given language $\mathcal{L}(\mathcal{N}')$ generated by a Petri net $\mathcal{N}'$ is included in the language $\mathcal{L}(\mathcal{N})$ generated by a *weakly deterministic* Petri net $\mathcal{N}$.

In the next step, we incorporate modalities in our approach and consider *modal Petri nets* with *may*- and *must*-transitions. For the definition of refinement, we follow again a language-based approach and use modal Petri nets as a device to generate modal language specifications. A subtle aspect of this generation concerns the treatment of silent *may*- and *must*-transitions. Refinement of modal Petri nets is then considered as refinement of their generated modal language specifications in the sense of Raclet *et al.* [25]. In particular, the *must* modality is important here to express that certain activities must be respected by refinements and hence implementations. On the other hand, the language inclusion property following from the *may* modality guarantess, as in the non modal case, that safety properties are preserved by refinements for all observable execution traces. We also extend the notion of weak determinacy to the modal case and show that the following problems, which extend the ones from above to the modal context, are decidable:

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 $\mathcal{S}(\mathcal{M})$.

As a particular important application scenario of our approach, we consider asynchronously communicating Petri nets. To realize the communication abilities we distinguish between input, output and internal labels which leads to our notion of a *modal I/O-Petri net*. We define an asynchronous composition operator for such nets where all actions related to communication (i.e. sending and receiving) are represented by internal labels. I/O-Petri nets obtained by asynchronous composition typically exhibit infinite state spaces. When considering refinements, transitions with internal labels should be treated as silent transitions. Therefore, we introduce a hiding operator on I/O-Petri nets which relabels internal labels to the silent label ϵ. Then we can directly apply our techniques and decidability results described above for the treatment of refinements in the context of asynchronously composed I/O-Petri nets. Thus our machinery is particularly useful in typical situations where an infinite abstract requirement specification is implemented by an architecture consisting of asynchronously communicating components. We will illustrate such an application in the context of a cash desk system case study.

Outline of the paper. We proceed by reviewing in Sect. 2 modal language specifications and their associated notion of refinement. Sect. 3 consists of two parts: In Sect. 3.1 we first recall the notion of a Petri net with silent transitions and then we define its generated language specification by abstracting silent transitions away. We also introduce the notion of weak determinacy and state the first two decision problems from above. In Sect. 3.2 we extend our approach to modal Petri nets. We define the generated modal language specification of a modal Petri net, consider modally weakly determinsitic Petri nets and state the last two decsion problems from above in the context of modalities. In Sect. 4, we focus on asynchronously communicating modal Petri nets over an I/O-alphabet with distinguished input, output, and internal labels. We define the asynchronous composition of modal I/O-Petri nets and adopt them, by hiding of internal labels, to our approch for modal refinement. Then, in Sect. 5, we discuss a case study to illustrate our principles. In Sect. 6 we present the decision algorithms of the four decision problems mentioned above. Sect. 7 concludes this paper and presents some future work perspectives.

2 Modal Language Specifications

Modal specifications were introduced by Larsen and Thomsen in [18] in terms of modal transition systems where transitions are equipped with distinguished *may* (allowed) and *must* (required) modalities. This idea has been adapted by Raclet in his (French) Ph.D. thesis [23] in which he applied it to language specifications where the complexity of decision problems is more tractable than with modal transition systems. Moreover, modal refinement is sound and complete with the language-based formalism while it is non-complete with the transition system based formalism [16]. Therefore we base our considerations on a language approach to modal specifications. Let us first review the underlying definitions of modal language specifications and their refinement as introduced in [24].

Notation. Let E be a set, then $P(E)$ denotes its powerset.

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*,*
- $a \notin may(u)$ *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) $\forall\ u \in \mathcal{L}, may(u) = \{a \in \Sigma \mid u.a \in \mathcal{L}\}$

Observation. 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 $must(u)$.

Example 1. Let us consider the example of a message producer and a message consumer represented in Fig. 1. The transition with label *in* represents an input received by the producer from the environment, which is followed by the transition with label m representing the sending of message m. The consumer must be able to perform the transition labeld with m, representing the reception of m, and then it must be able to "output" *out* to the environment.

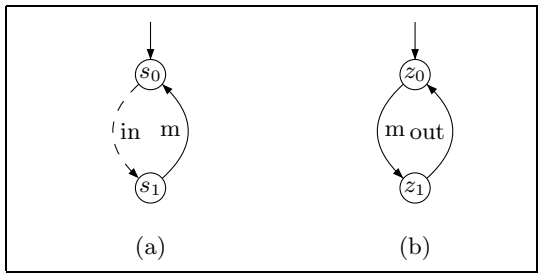

Fig. 1. Modal transition systems for a producer (a) and a consumer (b)

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 (solid line). In the consumer model all transitions are required. The language associated with 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:

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

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 can be refined by either removing some allowed events or changing them into required events.

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 $\mathcal{L}'$.*

Modal language specifications support underspecification and thus stepwise refinement but they are not appropriate for the specification of infinite state systems since refinement can not be decided in this case. Moreover, they do not support silent transitions and hence observational abstraction.

3 Modal Petri Nets

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. Silent transitions are important to model, e.g., internal alternatives which are invisible to the outside. In the following of this section we will consider such Petri nets and extend them by modalities.

3.1 Petri Nets and their Generated Languages

In this subsection, we first review basic definitions of Petri net theory and then we develop our first decision problems without taking into account modalities yet.

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 $\mathbb{N}$; it 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 $m : P \mapsto \mathbb{N}$. A transition $t \in T$ is called *silent*, if $\lambda(t) = \epsilon$. The labeling function is extended to sequences of transitions $\sigma = t_1 t_2 ... t_n \in T^*$ where $\lambda(\sigma) = \lambda(t_1)\lambda(t_2)...\lambda(t_n)$. According to this definition $\lambda(\sigma) \in \Sigma^*$, i.e. $\lambda(\sigma)$ represents a sequence of observable actions where silent transitions are abstracted away.

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 rest of the paper, 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 notions related to the semantics of a net.

Definition 4 (Firing rule). *Let $\mathcal{N}$ be a Petri net. A transition $t \in T$ is firable from 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 from 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 $Reach(\mathcal{N}, m_0) = \{m \mid \exists \sigma \in T^* such\ that\ m_0[\sigma\rangle m\}$.*

The reachable markings of a Petri net correspond to the reachable states of the modeled system. Since the capacity of places is unlimited, the set of the reachable markings of the Petri nets considered here may be infinite. Thus, Petri nets can model infinite state systems.

The semantics of a net will be given in terms of its generated language. It is important to note that the language generated by a labeled Petri net abstracts from silent transitions and therefore determines the observable execution traces of a net. Technically this is achieved by using the empty word ϵ as a label for silent transitions such that, for each sequence $\sigma \in T^*$ of transitions, $\lambda(\sigma)$ shows only the labels of the observable transitions.

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 [21]. In our approach, however, we deal with silent transitions which often model non-deterministic choices. For instance, silent transitions with non-deterministic choices will naturally appear in asynchronous compositions considered later on (c.f. Sect. 4) when communication actions are hidden. Therefore, we are interested in a relaxed notion of determinacy which allows us to consider determinism "up to silent moves". 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 both be extended to produce the same continuations of u. In other words, in a weakly deterministic Petri net, one may fire from a reachable marking two different sequences labelled by the same word but the visible behaviours from the reached markings are the same. So an external observer cannot detect this non-determinism. In this sense our notion of weak deterministic Petri net corresponds to Milner's (weak) determinacy [20] and to the concept of a weakly deterministic transition system described in [11]. The underlying idea is also related to the notion of output-determinacy introduced in [14].

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*

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

be the set of labels that may occur starting from m after a sequence of silent transitions. $\mathcal{N}$ is called weakly deterministic*, if for each $\sigma, \sigma' \in T^*$ with $\lambda(\sigma) = \lambda(\sigma')$ and for markings m and m' with $m_0[\sigma\rangle m$ and $m_0[\sigma'\rangle m'$, we have $may_{mk}(m) = may_{mk}(m')$.*

Weakly deterministic Petri nets will play an important role for deciding refinements. Hence, 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 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 [10]. However, in [21] 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 on 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 $\mathcal{L}(\mathcal{N})$.

The descion problems deal with refinements based on an observational abstraction of silent transitions of the underlying Petri nets.

3.2 Modal Petri Nets

In the following we extend our approach by incorporating modalities. For this purpose we introduce, following the ideas of modal transition systems and modal language specifications, modal Petri nets with modalities *may* and *must* on their transitions. Note that we require any *must*-transition to be allowed, i.e. to be a *may*-transition as well.

Definition 8 (Modal Petri net). *A* modal Petri net $\mathcal{M}$ *over an alphabet Σ is a pair $\mathcal{M} = (\mathcal{N}, T_\Box)$ where $\mathcal{N} = (P, T, W^-, W^+, \lambda, m_0)$ is a labeled Petri net over Σ and $T_\Box \subseteq T$ 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 producer may receive a message *in* (white transition) but must produce a message m (black transition). The consumer must receive a message m and then it must produce a message *out*.

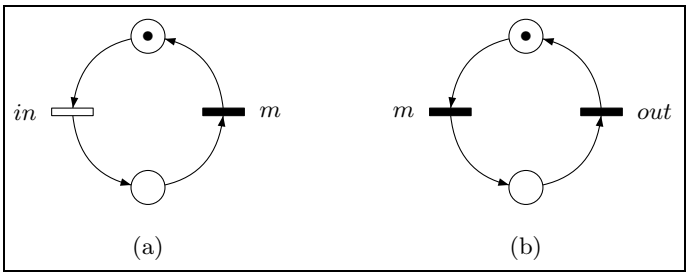

Fig. 2. Modal Petri nets for a producer (a) and a consumer (b)

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 of the generated language specification should take into account abstraction from silent transitions. While for the *may* modality this is rather straightforward, the definition of the *must* modality is rather subtle, since it must take into account silent *must*-transitions which are abstracted away during language generation. For the definition of the *must* modality we introduce the following auxiliary definition which describes, for each marking m, the set $must_{mk}(m)$ of all labels $a \in \Sigma$ which must be produced by firing (from m) some silent *must*-transitions succeeded by a *must*-transition labeled by a. This means that the label a must be producible as the next visible label by some firing sequence of m. Formally, for any marking m, let

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

We can now consider 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[1], the set $must_{mk}(m)$. The labels in $must_{mk}(m)$ must be exactly the possible successors 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:*

[1] We require this to avoid false detection of must transitions starting from intermediate markings which are reachable by silent may transitions.

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

Remark 1. Any modal language specification generated by a modal Petri net is consistent. Condition $C1$ is a consequence of the inclusion $must_{mk}(m) \subseteq may_{mk}(m)$ and condition $C2$ is a consequence of the definition of $may_{mk}(m)$.

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

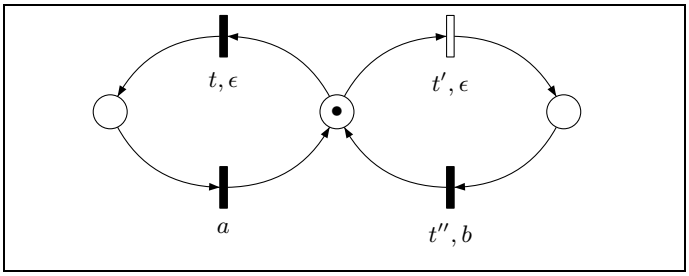

Fig. 3. Modal Petri net with silent transitions

The modal language specification generated by this net consists of the language $\mathcal{L}$ presented by the regular expression $(a^*b^*)^*$ and 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* $\mathcal{N}$*.* $\mathcal{M}$ *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. The Petri net in Fig. 3 (considered without modalities) is not weakly deterministic. Indeed let m_1 be the marking reached by t from m_0. Both markings m_0 and m_1 are reachable by a sequence of silent transitions. However from m_0, the sequence $t't''$ labelled by b is fireable while no sequence labelled by b is fireable from m_1. Hence it is also not modally weakly deterministic.

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

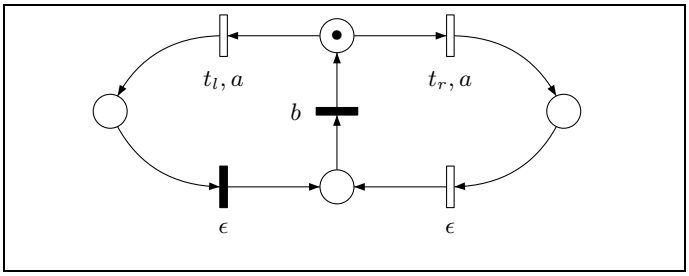

Fig. 4. Non weakly deterministic modal Petri net

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 $must_{mk}(m_l) = \{b\}$ while $must_{mk}(m_r) = \emptyset$ (since the silent transition firable in m_r is only a *may*-transition). Hence the Petri net is not modally weakly deterministic. However, one should note that, if we forget the modalities, then the Petri net in Fig. 4 is weakly deterministic.

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 and Asynchronous Composition

A particular application scenario for our approach is given by systems of asynchronously communication components. The characteristic property of this communication style is that communication happens via a potentially unbounded event channel such that the actions of sending (output) and receiving (input) of a message are delayed. In order to model such systems 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 $\Sigma = in \uplus out \uplus int$. Since $\epsilon \notin \Sigma$, this means that all labels, including the internal ones, are observable. 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 mean 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 [2, 17], 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 $\Sigma_1 \cap \Sigma_2 = (in_1 \cap out_2) \cup (in_2 \cap out_1)$.[2]

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:*

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

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 $a^{\triangleright}$ and ${}^{\triangleright}a$ where the former represents the asynchronous sending of a message and the latter represents the receipt of the message (at some later point in time).

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

We are now able to define the asynchronous composition of composable modal I/O-Petri nets. We first take the disjoint union of the two nets. Then we add a new place p_a (called channel place) for each shared label a.[3] The shared label a of every output transition t becomes $a^{\triangleright}$ and such a transition produces a token in p_a. The shared label a of every input transition t becomes ${}^{\triangleright}a$ and such a transition consumes a token from p_a. The next definition formalizes this description.

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 $p \in P_1 \cup P_2,\ t \in T_c$,*

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

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

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

$$W_c^-(p_a,t) = \begin{cases} 1 & \text{if } a = \lambda_i(t) \in in_i \cap out_j \\ 0 & \text{otherwise} \end{cases}$$

$$W_c^+(p_a,t) = \begin{cases} 1 & \text{if } a = \lambda_i(t) \in in_j \cap out_i \\ 0 & \text{otherwise} \end{cases}$$

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

$$\lambda_c(t) = \begin{cases} \lambda_i(t) & \text{if } t \in T_i,\ \lambda_i(t) \notin \Sigma_1 \cap \Sigma_2 \\ {}^{\triangleright}\lambda_i(t) & \text{if } t \in T_i,\ \lambda_i(t) \in in_i \cap out_j \\ \lambda_i(t)^{\triangleright} & \text{if } t \in T_i,\ \lambda_i(t) \in in_j \cap out_i \end{cases}$$

[3] Technically, the new places p_a play the same role as interface places in open nets. However, the "openness" of a modal I/O-Petri net is determined by the input/output lables on its transitions rather than by input/output places.

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

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

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

Proof. (Sketch) We introduce the following notations. Given a word σ over some alphabet A and $A' \subseteq A$, $|\sigma|$ denotes the length of σ, $\sigma_{\lfloor A'}$ denotes the projection of σ on A' and $|\sigma|_{A'}$ is defined by $|\sigma|_{A'} = |\sigma_{\lfloor A'}|$. Given a shared label a, $out(a)$ ($in(a)$ resp.) denotes the set of output (input resp.) transitions with label a. Given a marking m of $\mathcal{M}_c$, $m^{(i)}$ denotes the projection of m on P_i with $i \in \{1, 2\}$.

We start the proof with a key observation. Let $m_0[\sigma\rangle m$ be a firing sequence of $\mathcal{M}_c$. Then:

– For every $i \in \{1, 2\}$, $m_{i_0}[\sigma_{\lfloor T_i}\rangle m^{(i)}$ is a firing sequence of $\mathcal{M}_i$.
– For every shared label a, $m(p_a) = |\sigma|_{out(a)} - |\sigma|_{in(a)}$.

We only prove that $\mathcal{M}_c$ is weakly deterministic. The proof of the other condition is similar.
Let $m_0[\sigma\rangle m[\sigma^* t\rangle m^*$ and $m_0[\sigma'\rangle m'$ be a firing sequence of $\mathcal{M}_c$ with $\lambda_c(\sigma) = \lambda_c(\sigma')$, $\lambda_c(\sigma^*) = \epsilon$ and $\lambda_c(t) \in \Sigma_c$. Due to the equality of labels of σ and σ' (and using the key observation), for every $i \in \{1, 2\}$, $m^{(i)}$ and $m'^{(i)}$ are reached by sequences with same labels and for every shared label a, $m(p_a) = m'(p_a)$.

Assume w.l.o.g. that $t \in T_1$, then by weak determinism of $\mathcal{M}_1$ there is a firing sequence $m'^{(1)}[\sigma_1'^*\rangle$ with $\lambda_1(\sigma_1'^*) = \lambda_1(t)$. If t is not an input transition there is no input transition in $\sigma_1'^*$. So $\sigma_1'^*$ is fireable from m' in $\mathcal{M}_c$ and we are done. Otherwise let $a = \lambda_1(t)$ then (1) $m(p_a)(= m'(p_a)) > 0$ and (2) there is exactly an input transition in $\sigma_1'^*$ and its label is a. So again $\sigma_1'^*$ is fireable from m' in $\mathcal{M}_c$. □

Example 5. 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 $m^{\triangleright}$ and ${}^{\triangleright}m$. The Petri net composition describes an infinite state system and its generated modal language specification has a language which is no longer regular.

We will now discuss how the results of our theory presented in Sect. 3 can be applied in typical application scenarios involving asynchronous composition. Assume given an abstract requirements specification for a possibly infinite state system given in form of a modal I/O-Petri net $\mathcal{M}$. As an example for $\mathcal{M}$ consider

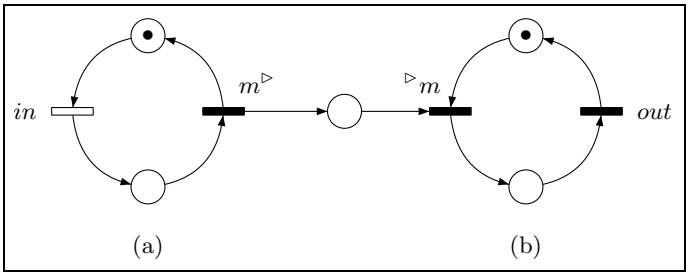

Fig. 5. Composition of the producer and consumer Petri nets

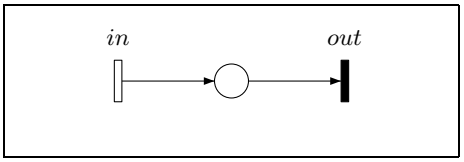

Fig. 6. Requirements specification for an infinite state producer/consumer system

the requirements specification for an infinite state producer/consumer system presented by the modal I/O-Petri net in Fig. 6. The system is infinite since the transition labeled with *in* can always fire. Such requirements specifications are typically implemented by an architecture which involves several communicating components, presented by modal I/O-Petri nets, say $\mathcal{M}_1$ and $\mathcal{M}_2$. Such components could be, for instance, off-the-shelf components which are now reused for a particular purpose. Then one is interested to check whether the designed architecture, say $\mathcal{M}_c = \mathcal{M}_1 \otimes_{as} \mathcal{M}_2$, obtained by component composition is indeed a correct implementation/refinement of the given requirements specification $\mathcal{M}$. An example architecture for a producer/consumer system is shown by the asynchronous composition of the two Petri nets in Fig. 5. According to our theory in Sect. 3, implementation correctness could mean that the modal language specification $\mathcal{S}(\mathcal{M}_c)$ generated by $\mathcal{M}_c$ is a modal language specification refinement, in the sense of Def. 2, of the modal language specification $\mathcal{S}(\mathcal{M})$ generated by $\mathcal{M}$. Unfortunatley, we can immediately detect that this cannot be the case since the asynchronous composition has additional communication actions represented by internal labels of the form $a^{\triangleright}$ and $^{\triangleright}a$ which are not present in the abstract requirements specification. In the example the internal labels are just the labels $m^{\triangleright}$ and $^{\triangleright}m$ used for the communication. On the other hand, from the observational point of view internal actions, in particular communication actions, can be abstracted away when considering refinements. In our approach this can be simply achieved by relabeling internal labels, and hence communication labels $a^{\triangleright}$ and $^{\triangleright}a$, to ϵ which is formally defined by the following hiding operator:

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

Coming back to the considerations from above we can then express implementation correctness by requiring that the modal language specification $\mathcal{S}(\alpha(\mathcal{M}_c))$ generated by $\alpha(\mathcal{M}_c)$ is a modal language specification refinement of the modal language specification $\mathcal{S}(\mathcal{M})$ generated by $\mathcal{M}$. According to the results of Sect. 6, we can decide this if $\mathcal{M}$ and $\alpha(\mathcal{M}_c)$ belong to the class of weakly deterministic modal Petri nets which again is decidable. Looking to the producer/consumer example the "abstract" Petri net in Fig. 6 is obviously modally weakly deterministic. For the composed Petri nets in Fig. 5 it is not obvious that the hiding operator produces a modally weakly deterministic net but, according to the results of Sect. 6, we can decide it (and get a positive answer). Note, however, that in the case where one of the transitions used for the communication were not a "must" then, after hiding, the Petri net composition would not satisfy the second condition of modal weak determinacy. Finally, we can also decide whether the refinement relation holds in the example between the generated modal language specifications and get also a positive answer.

5 Case Study: Cash Desk Application

In order to illustrate our approach with a more ambitious example we consider a cash desk application (inspired by the case study of a trading system in [26]). First, we provide a requirement specification for the behavior of the cash desk in terms of the modal I/O-Petri net shown in Fig. 7. The net is based on an I/O-alphabet partitioned into sets of input labels and output labels. There are no internal labels in this requirement specification. Here and in the following drawings we write $m?$ to indicate that a label m belongs to the set of input labels; similarly, we write $m!$ if a label m belongs to the set of output labels.

The requirement specification says that a cash desk must be able to read repeatedly an item, represented by the must-transition with input label *item?*, and that afterwards it must be able to print (on the bill) each item's details (output *printItem!*). Since the number of items to be processed is not limited, and since the cash desk can proceed with reading the next item before the details of the previous item have been printed, the system specification involves infinitely many states. On the other hand, the specification determines what must/may be done when a sale is finished. First, when no item is read anymore, it must be possible to finish a sale (input *finishSale?*) and then to print the total sum of the sale (output *printTotal!*). Afterwards the cash desk must be able to accept cash payment (input *cash?*), but it may also be implemented in such a way that one can choose a credit card for the payment. Hence, the transition with

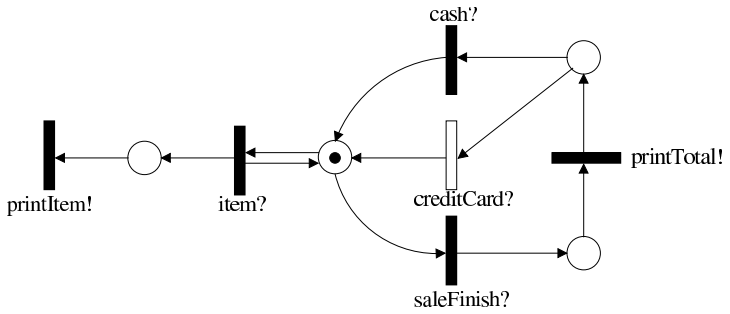

Fig. 7. Requirement specification for a cash desk

input label *creditCard?* is a may-transition. Note that *printItem!* may always interleave with other actions but at most as often as *item?* has occurred before. Hence this is a loose specification. In practice one would expect that *printTotal!* can only occur if the details of all inputed items are indeed printed before, but to specify such a property would need further extensions of the formalism.

We are now going to provide a refinement of the cash desk requirement specification. For this purpose we want to use two components, one representing a graphical user interface of the cash desk (Cash Desk GUI) and another one representing a controller of the cash desk (Cash Desk Controller). The single components with their input and output labels are shown in Fig. 8 and the behavior of each single component is specified by the modal I/O-Petri nets in Fig. 9.

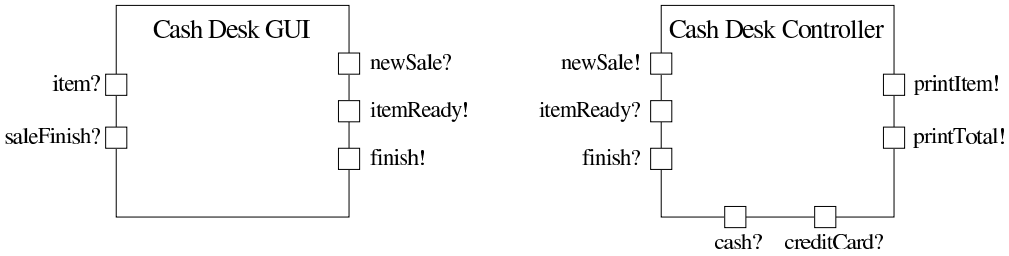

Fig. 8. Components Cash Desk GUI and Cash Desk controller

The specified behavior of the cash desk GUI is straightforward. It is initiated by an input of *newSale?* from the environment and then the GUI must be able to read iteratively items (input *item?*) or to accept a request to finish a sale (input *saleFinish?*). After an item has been read the GUI must immediately forward the details of the item for further processing to the environment (output *itemReady!*). Similarly, if the GUI has received a request to finish a sale it must forward this request to the environment to take over the control (output *finish!*).

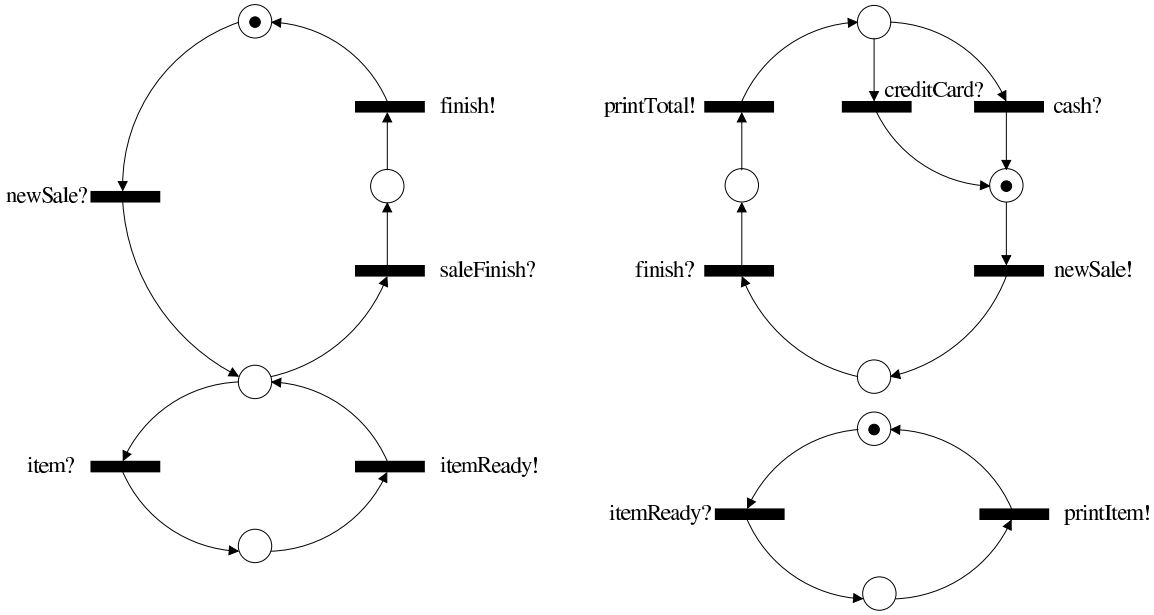

Fig. 9. Modal I/O-Petri nets for Cash Desk GUI (left) and Cash Desk Controller (right)

The behavior of the cash desk controller specified by the Petri net on the righthand side of Fig. 9 is more subtle. It contains two unconnected parts, the "upper" part describing a global protocol that a cash desk controller must follow and the "lower" part controlling the printing of items. As a consequence, the cash desk controller is always able to consume items with *itemReady?* and then to print the item with *printItem!*. In particular *printItem!* can interleave everywhere (which, in fact, is also possible in the requirement specification). Of course, this is not a desirable behavior for a concrete implementation of the cash desk controller and it would be better if the cash desk controller could only finish (*finish?*) if there are no more items provided by the environment for *itemReady?*. In a concrete implementation this could be achieved, e.g., by introducing a priority such that the cash desk controller priorizes available inputs for *itemReady?* against available inputs for *finish?*. Such an implementation would be conform to our specification of the cash desk controller, but to specify such a priorization would need further extensions of the formalism.

Finally, we connect the two components which yields the system architecture shown in Fig. 10 whose dynamics is described by the asynchronous composition of the I/O-Petri nets of the single components as shown in Fig. 11.[4]

[4] Remember that in the asynchronous composition of Petri nets the (internal) communication labels are either of the form $m^{\triangleright}$ (representing the sending of an output to the channel place) or of the form ${}^{\triangleright}m$ (representing the receiption of an input from the channel place).

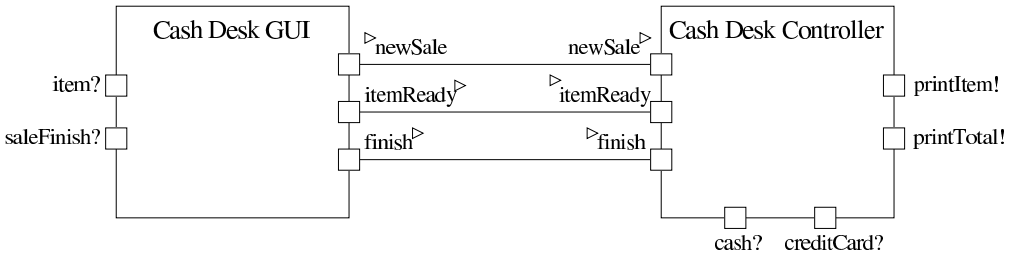

Fig. 10. Composition of cash desk GUI and cash desk controller components: static view

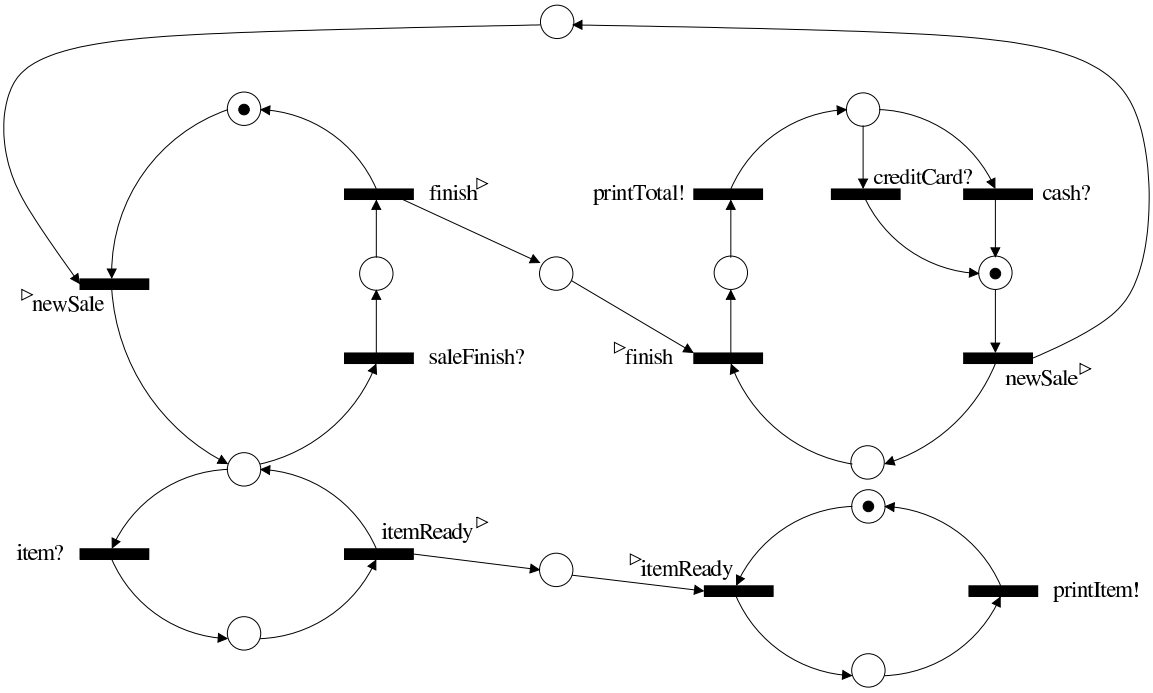

Fig. 11. Asynchronous composition of cash desk GUI and cash desk controller Petri nets

All Petri nets considered here are modally weakly determinisitic, which is decidable by the results of Sect. 6. Moreover, after hiding the communication labels of the form $m^{\triangleright}$ and ${}^{\triangleright}m$ in the asynchronous composition in Fig. 11 (by relabeling them to ϵ), the resulting net is also modally weakly deterministic. Therefore we can also decide, according to Sect. 6, whether the modal language specification generated by the Petri net obtained by hiding the internal communication labels from the asynchronous composition in Fig. 11 is a modal refinement of the modal language specification generated by the modal Petri net for the requirement specification in Fig. 7. This is indeed the case but requires a detailed proof.

6 Decision Algorithms

We begin this section by some recalling 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* [9] 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 [19]. Furthermore this procedure can be adapted to semi-linear sets when markings are identified to vectors of $\mathbb{N}^{|P|}$. Given a semi-linear set E of markings, in order to decide whether there exists a marking in 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 [28] 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 $\mathcal{N}'$ (1) whose places are two copies of places of $\mathcal{N}$, (2) whose firing sequences are pairs of firing sequences in $\mathcal{N}$ with the same label and, (3) whose markings are pairs of the corresponding reached markings in $\mathcal{N}$. Then we define the semilinear sets F_a with $a \in \Sigma$ that characterize the markings of $\mathcal{N}'$ where a firing sequence can produce the label a starting from the first item of the pair while such a sequence does not exist from the second item of the pair. Proceeding in such a way reduces the weak determinism $\mathcal{N}$ to a reachability problem for $\mathcal{N}'$.

Let us detail the proof. $\mathcal{N}'$ is 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 firable after the firing of silent transitions (using results of [28]) 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 $\mathcal{N}''$ (1) whose places are copies of places of $\mathcal{N}$ and $\mathcal{N}'$, (2) whose firing sequences are pairs of firing sequences in $\mathcal{N}$ and $\mathcal{N}'$ with the same label and, (3) whose markings are pairs of the corresponding reached markings in $\mathcal{N}$ and in $\mathcal{N}'$. Then we define the semilinear sets F_a with $a \in \Sigma$ that characterize the markings of $\mathcal{N}''$ where a firing sequence can produce the label a starting from the item of the pair corresponding to $\mathcal{N}'$ while such a sequence does not exist from the item of the pair corresponding to $\mathcal{N}$. Now the key observation is that since $\mathcal{N}$ is weakly deterministic, from any other marking of $\mathcal{N}$ reached by a sequence with the same label there does not exist such a sequence. Thus proceeding in such a way reduces the inclusion $\mathcal{L}(\mathcal{N}') \subseteq \mathcal{L}(\mathcal{N})$ to a reachability problem for $\mathcal{N}''$.

Let us detail the proof. The net $\mathcal{N}''$ is 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 transition of $\mathcal{N}''$.
– The initial marking is $m_0 + m'_0$.

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 firable 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. We first recall the two conditions for weak determinism:

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 $must_{mk}(m) = must_{mk}(m')$.

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 firable 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. We first recall the two conditions for refinement:

1. $\mathcal{L}' \subseteq \mathcal{L}$,
2. 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 $\mathcal{L}'$.

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 $\overline{G'_a}$. □

7 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. It has been shown that our theory is particularly useful in the context of refinements given by an architecture of asynchronously communicating components.

Our work extends the expressive power of modal transition systems [18, 17] and of modal language specifications [23, 24] since Petri nets allow to specify with a finite representation the behavior of *infinite* state systems. On the other hand, our work provides also an extension of Petri net theory since we have considered *observational* refinement abstracting silent moves during language generation and we have introduced *modalities* for Petri nets. In particular, the *must* modality is important to express that certain activities must be respected by refinements and hence implementations.

An important role in our approach has been played by the hypothesis of weak determinacy which has two justifications. First, without this hypothesis, the refinement problem is undecidable (but we have not proved it in this paper). Secondly, as nets represent specifications, using weakly deterministic Petri nets limits the class of specifications, but this class is already much more expressive than regular specifications (the standard model). But still weak determinacy is a proper restriction since it does not allow internal decisions if they lead to different observable execution traces. More discussions and examples of weak determinacy in the context of state transition systems without modalities can be found in [11]. In particular, it is shown in [11] that a transition system is weakly deterministic if its minimization w.r.t weak bisimulation leads to a deterministic transition system without silent transitions. From the point of view of complexity, it is well known that reachability of Petri nets has a high theoretical complexity but in practical cases of specifications the empirical complexity is significantly smaller than the theoretical one.

For the application of our theory to asynchronously communicating components we have used a hiding operator which translates communication actions into silent transitions (which then are abstracted for refinement). Since, in general, the hiding operator does not preserve modal weak determinacy, we are interested in the investigation of conditions which ensure this preservation prop-

erty. This concerns also conditions for single components such that the hiding of communication labels from their composition is automatically modally weakly deterministic. We claim that this is indeed the case if the single components are modally weakly determinsitic and if all transitions concerning communication actions are *must*-transitions. Another direction of future research concerns the study of behavioral compatibility of interacting components on the basis of modal I/O-Petri nets, and the establishment of an interface theory for this framework along the lines of [1, 17, 5]. This would involve, in particular, compositionality results on the preservation of local refinements by global compositions. Such results are important to achieve independent implementability and thus substitutability of components which may lead to applications of our theory to web service substitutability as considered in [27]. Actually, the refinement notion proposed in [27], called *accordance* there, is conceptually different from ours. In [27], refinement is motivated by requiring preservation of interaction correctness of services with their communication partners. In our approach we do not consider communication partners for the refinement definition, but we compare behaviors of abstract and concrete modal Petri nets (formalized by their generated modal language specifications). Then refinement ensures, first, that all observable execution traces of the concrete net are allowed by the abstract one, hence safety properties are preserved, and secondly, that all required transitions of the abstract net are respected by the concrete one. In modal interface theories [17, 5, 12] this property has then the *consequence* that certain required interactions are preserved by refinements, much in the spirit of [27].

Acknowledgement. We are very grateful to all reviewers of the submitted version of this paper who have provided detailed reports with many valuable hints and suggestions.

References

1. L. de Alfaro and T. A. Henzinger. Interface Theories for Component-Based Design. In Proc. EMSOFT'01, pages 148–165, 2001.
2. L. de Alfaro and T. A. Henzinger. Interface-based Design. In NATO Science Series: Mathematics, Physics, and Chemistry, vol. 195, pages 83–104, Springer, 2005.
3. 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.
4. 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.
5. 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), vol. 6015 of *LNCS*, pages 175–189, Springer, 2010.
6. N. Bertrand, A. Legay, S. Pinchinat, J-B. Raclet A Compositional Approach on Modal Specifications for Timed Systems. *ICFEM 2009*, pages 679–697, 2009.

7. D. Brand and P. Zafiropulo On communicating finite-state machines. *JACM*, volume 30(2), pages 323–342, 1983.
8. B. Delahaye, K. G. Larsen, A. Legay, M. L. Pedersen, and A. Wasowski Decision Problems for Interval Markov Chains *LATA'2011*, to appear
9. S. Ginsburg and E. H. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2) pages 285–296, 1966.
10. M.H.T. Hack. Decidability questions for Petri Nets. Ph.D.Thesis. M.I.T (1976)
11. R. Hennicker and S. Janisch and A. Knapp. On the Observable Behaviour of Composite Components. In Electr. Notes Theor. Comput. Sci. Volume 260, pages 125–153, 2010.
12. R. Hennicker and A.Knapp. Modal Interface Theories for Communication-Safe Component Assemblies. In Proc. 8^{th} Int. Conf. on Theoretical Aspects of Computing (ICTAC'11), *LNCS*, Springer, 2011, to appear.
13. R.M. Karp, R.E. Miller Parallel program schemata. In: *JTSS* 4, 1969, pages 147–195.
14. V. Khomenko, M. Schaefer and W. Vogler. Output-Determinacy and Asynchronous Circuit Synthesis. In: Fundam. Inf. 88, 4, pages 541–579, 2008.
15. K.G. Larsen. Modal specifications. In: *Joseph Sifakis, ed., Automatic Verification Methods for Finite State Systems*, vol. 407 of LNCS, pages 232–246, 1989.
16. 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.
17. K.G. Larsen, U. Nyman and A. Wasowski. Modal I/O automata for interface and product line theories. In *Prog. Languages and Systems, 16th European Symposium on Programming (ESOP'07)*, vol. 4421 of *LNCS*, pages 64–79. Springer, 2007.
18. 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.
19. E. Mayr. An algorithm for the general Petri net reachability problem. In *Proc. 13th Annual ACM Symp. on Theor. of Computing (STOC'81)*, pages 238–246, 1981.
20. R. Milner Communication and concurrency. Prentice Hall, 1989.
21. E. Pelz. Closure properties of deterministic Petri net languages. In *STACS'87*, vol. 247 of *LNCS*, Springer 1987.
22. J.L. Peterson, Petri net theory and the modeling of systems, Prentice-Hall, Englewood Cliffs, NJ, 1981
23. J.-B. Raclet. Quotient de spécifications pour la réutilisation de composants. *PhD Thesis*, Ecole doctorale Matisse, Université de Rennes 1, IRISA. December 2007
24. 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.
25. 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.
26. A. Rausch, R. Reussner, R. Mirandola, U. Nyman and F. Plasil. The Common Component Modeling Example: Comparing Software Component Models. vol. 5153 of *LNCS*, Springer, 2008.
27. C. Stahl, K. Wolf. Deciding service composition and substitutability using extended operating guidelines. *Data Knowl. Eng.*, 68(9): 819-833 (2009).
28. 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, pages 234–258, 1984.