Modal Open Petri Nets

Vitali Schneider and Walter Vogler(✉)

Institut für Informatik, University of Augsburg, Augsburg, Germany
walter.vogler@informatik.uni-augsburg.de

Abstract. Open nets have an interface of input and output places for modelling asynchronous communication; these places serve as channels when open nets are composed. We study a variant that inherits modalities from Larsen's modal transition systems. Instantiating a framework for open nets we have developed in the past, we present a refinement preorder in the spirit of modal refinement. The preorder supports modular reasoning since it is a precongruence, and we justify it by a coarsest-precongruence result. We compare our approach to the one of Haddad et al., which considers a restricted class of nets and a stricter refinement. Our studies are conducted in an extended class of nets, which additionally have transition labels for synchronous communication.

1 Introduction

On an abstract level, concurrent systems can be specified and developed with the well-known labelled transition systems (LTS). The labels of such an LTS are the actions of the system, including the *hidden* action τ. To combine components to larger systems according to synchronous communication, parallel composition $\|$ merges equally-labelled transitions of two components; one might also hide such labels. Furthermore, a relation for stepwise refinement is needed that supports modular reasoning: if one refines a component of a parallel composition, then this should result in a refinement of the overall system. A refinement relation with this property is called a *precongruence* w.r.t. $\|$.

Such a precongruence can be defined as inclusion of the LTS-languages or some other trace-based semantics, or it can be some kind of bisimilarity, see [7] for an overview. These refinement relations can easily be transferred to (labelled) Petri nets, cf. e.g. [14,21], with precongruence results for an analogous parallel composition. Advantages of Petri nets are that they are distributed by nature as are concurrent systems, and that they can give a finite representation for infinite state systems.

Bisimilarity allows one, in particular, to refine an LTS to a parallel composition with new hidden transitions resulting from communication. Such a composition can be a step forward to an implementation. But bisimilarity, being an

Research support provided by the DFG (German Research Foundation) under grant no. VO 615/12-2.

S. Donatelli and S. Haar (Eds.): PETRI NETS 2019, LNCS 11522, pp. 25–46, 2019.
https://doi.org/10.1007/978-3-030-21571-2_2

equivalence, does not offer much leeway and is, thus, in general not so appropriate for refinement.

Modal transition systems (MTS) are a ground breaking improvement towards *loose specifications* [11]. An MTS is an LTS with two kinds of transitions: *must*-transitions are required, while *may*-transitions are allowed, but only optional. A *modal refinement relation* can be described as an alternating simulation: each must-transition of the *specification* has to be simulated by an equally labelled must-transition of the refinement – possibly using additional hidden must-transitions; analogously, a may-transition of the *refinement* has to be allowed in the specification by a number of may-transitions. Also modalities and modal refinement can be transferred to Petri nets, see the discussion of modal Petri nets (MPN) below.

Petri nets are particularly well suited for modelling asynchronous communication, where the sender of a message does not have to wait for the receiver. If the order of messages on the same channel is not relevant, one can simply connect sending transitions via a (channel) place to receiving transitions instead of merging transitions as in the synchronous case. For such a setting, we model systems as so called *open* nets,[1] which have an interface consisting of two disjoint sets of special input and output places.

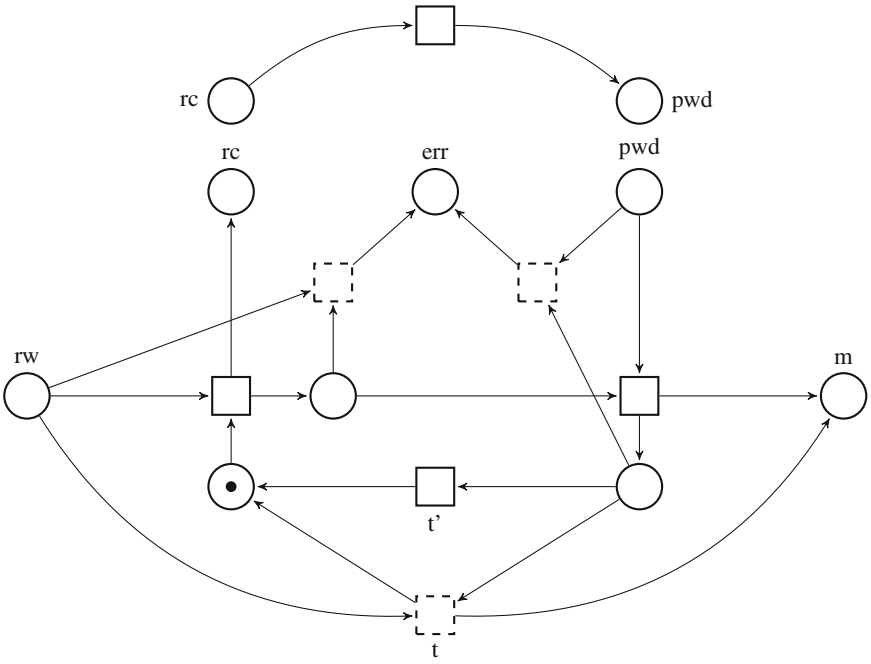

Fig. 1. Modal open nets *ATM* and password provider

[1] The term open in this sense presumably stems from [1].

As an example, consider the lower net *ATM* in Fig. 1; this is actually a modal open net (MON), where the dashed boxes denote may-transitions while the others denote must-transitions. The places *rw* and *pwd* are input places (no ingoing arcs from the net), the other named places are output places. If a user of the modelled system puts a token onto *rw*, this **r**equests to **w**ithdraw some money. Initially, the system **r**equests additional **c**redentials by putting a token onto *rc*. After receiving a password via *pwd*, **m**oney is handed out on *m*. Now the system can decide to repeat this, but it optionally can do without asking for a password. In the latter case, a password is needed for the request after. The system can also be implemented with some ability to detect errors: an error message can be sent via place *err* if a withdrawal request is sent while the previous request is being handled or if a password is sent already when the previous interaction has just been finished.

The upper net describes a piece of software providing the user's password whenever it is required. Asynchronous composition $\oplus$ merges interface places with the same name, removing the name from the new interface. In the example, after installing the software and saving her password, the user deals with the composed system and does not have to enter her password again.

An early paper considering the composition of place-bordered nets as the above (without modalities) is [16], where nets are built composing deterministic nets. It is shown how to check (Petri net) liveness for such compositions. In [20], refining a transition t means to replace t (via place merging) by a so-called daughter-net whose border (interface) consists of the places incident to t. Results are given for which daughter-nets the replacement preserves behaviour like liveness and boundedness. Results on liveness for asynchronous compositions of more general nets can be found in [18]. A compositional semantics building Petri-net processes by place merging is presented in [10].

Extending [20], a general framework is suggested in [21] how to transfer semantics and refinement relations from a synchronous setting to open nets in a "sensible way". This is worked out e.g. in [19], which is one of a number of papers on open nets and operating guidelines like [12]. It is also applied in [2] in a setting with labelled open Petri nets; it is shown that some variants of bisimilarity are congruences for one operator that combines $\|$ and $\oplus$. By a "sensible" refinement relation we mean a relation that accepts a refinement unless there is a formal reason against this. One general formal requirement is that the relation supports modular reasoning by being a precongruence. Additionally, one chooses some behavioural requirements; in [19] for example, it is required that a refinement step does not introduce a deadlock. Two trace sets are defined, and inclusion of these is shown to be sensible in the above sense, since it is the *coarsest* precongruence with the latter requirement.

In the present paper, we demonstrate that the above framework works also in a setting with modalities and alternating simulation. Although we are mainly interested in MON, we conduct our study for the larger class of *labelled MON* (ℓMON), which also have action-labelled transitions and include MON and MPN. The main contribution is a kind of modal refinement, which we show to be a precongruence for $\|$ and the coarsest precongruence for $\oplus$ respecting modal refinement on MPN.

MPN have been introduced in [6] in combination with a modal language as in [15]. The main issue is to decide weak determinism (a variant of determinacy as in [13]) and – for weakly deterministic, but possibly unbounded MPN – modal-language inclusion. Asynchronous composition is only defined to build large MPN, which are often unbounded due to the channel places. There are no precongruence results.

The net class and the composition from [6] are further studied in [8], which is very close to the present paper. There, the nets are called modal asynchronous I/O-Petri nets (MAIOPN or, as we write here, MAP); they are a different representation for a restricted class of MON: the nets are actually MPN and the interface places are left implicit. The places only become explicit during the composition $\oplus_{HH}$. The refinement relation is modal refinement and is shown to be a precongruence. Due to the special interests in [8], a MAP may have so-called *internal* actions, showing how the generated channel places are accessed. They are not really visible but still taken into account in modal refinement. As a consequence, a MAP can only be refined by another one, if the latter has the same channel places; one cannot refine a monolithic specification by an asynchronous composition. This is noted in [6,8], so for stepwise refinement it is suggested to hide the internal actions at the end of composition. With this modification, one can translate MAP into MON such that composition is preserved, i.e. MAP can be seen as a sub-setting of our setting.

We show that our modal refinement is coarser than the one on MAP, i.e. it is better from our perspective. One difference concerns a typical feature of asynchronous communication: if two messages are sent on different channels one after the other, there can be overtaking such that the environment cannot observe the order of sending. Hence, this order should not matter for the refinement relation. This is indeed the case in our approach, but it does matter in [8]. We also give an alternative proof for the MAP precongruence result, which we believe to be simpler conceptually. This paper revises and generalizes [17].

Section 2 introduces MTS and transfers parallel composition and hiding to MPN; the latter is also done in [8], but the actual net variant there is more complicated and an MTS (!) variant with additional Petri net places is used. Section 3 defines asynchronous composition and our refinement relation, pointing out that it preserves some liveness notion. The coarsest-precongruence for $\oplus$ and precongruence for the MTS operators are shown. Section 4 compares ours to the MAP-approach. The paper ends with a sketch how to restrict our approach to bounded nets in Sect. 5 and with some conclusions. We thank Alexander Knapp and Ayleen Schinko for supporting us with the figures, and the reviewers for their helpful comments.

2 Preliminaries

This section provides some basic notation for modal transition systems and modal Petri nets. Refinement and basic operations such as parallel composition, relabelling and hiding are transferred from MTS to MPN. The same holds for the precongruence results provided by Hüttel and Larsen in [9].

Most of the structures in this paper have an action *alphabet*, usually denoted by Σ. There is one hidden or invisible action τ, which is never in an alphabet. We denote $\Sigma \cup \{\tau\}$ by Σ^τ; a and α often stand for a typical action in Σ and Σ^τ resp. $\mathbb{N}$ denotes the set of natural numbers including zero.

2.1 Modal Transition Systems

In the introduction, we have already explained that MTS [11] have required must- and optional may-transitions. The condition $\longrightarrow \subseteq \dashrightarrow$ below reflects that every required transition should also be allowed.

Definition 1 (MTS). A *modal transition system (MTS)* is a tuple $Q = (S, \Sigma, \dashrightarrow, \longrightarrow, s^0)$ where S is a set of *states* containing the *initial state* s_0; Σ is an alphabet,

- $\dashrightarrow \subseteq S \times \Sigma^\tau \times S$ is the set of *may-transitions*, and
- $\longrightarrow \subseteq S \times \Sigma^\tau \times S$ is the set of *must-transitions* satisfying $\longrightarrow \subseteq \dashrightarrow$. $\Diamond$

We add the name of the MTS as an index to the components when needed or use e.g. S_i for the state set of Q_i etc., and similarly for nets later on. We write $s \overset{\alpha}{\dashrightarrow} s'$ for $(s, \alpha, s') \in \dashrightarrow$, and extend this to words $w \in (\Sigma^\tau)^*$: $s \overset{w}{\dashrightarrow} s'$ means that there is a sequence $s \overset{\alpha_1}{\dashrightarrow} s_1 \overset{\alpha_2}{\dashrightarrow} s_2 \ldots s_{n-1} \overset{\alpha_n}{\dashrightarrow} s'$ with $w = \alpha_1 \ldots \alpha_n$. Let $\widehat{w}$ be obtained from w by *removing all* τs. With this, we define the weak may-transition $s \overset{w}{=\!\Rightarrow} s'$ as $\exists v \in (\Sigma^\tau)^* : \widehat{v} = w \wedge s \overset{v}{\dashrightarrow} s'$. We have the analogous notations for must-transitions, writing $\Longrightarrow$ for $=\!\Rightarrow$. A state s is *reachable* in Q when $s^0 \overset{w}{\dashrightarrow} s$ for some $w \in \Sigma^\tau$.

The following defines the standard (weak) modal refinement for MTS as explained in the introduction.

Definition 2 (MTS refinement). Let Q_1 and Q_2 be two MTS over the same alphabet. We say that Q_1 is a *(modal)refinement* of Q_2, written $Q_1 \sqsubseteq_{MTS} Q_2$, if there exists an *MTS-relation* $\mathfrak{R} \subseteq S_1 \times S_2$ with $(s_1^0, s_2^0) \in \mathfrak{R}$ such that for every $(s_1, s_2) \in \mathfrak{R}$:

- $s_2 \xrightarrow{\alpha} s_2' \;\Rightarrow\; s_1 \overset{\widehat{\alpha}}{\Longrightarrow} s_1' \;\wedge\; (s_1', s_2') \in \mathfrak{R}$ and
- $s_1 \overset{\alpha}{\dashrightarrow} s_1' \;\Rightarrow\; s_2 \overset{\widehat{\alpha}}{=\!\Rightarrow} s_2' \;\wedge\; (s_1', s_2') \in \mathfrak{R}$. $\Diamond$

Note that, for two *implementations* (MTS with coinciding may- and must-transitions), MTS-relations and weak bisimulations [13] are the same. Next we define the operations of relabelling, hiding, parallel composition and parallel composition with hiding.

Definition 3 (MTS relabelling, hiding). A *relabelling function* for an alphabet Σ (and for MTS and MPN below with this alphabet) is a surjective function $f : \Sigma \to \Sigma'$; additionally, we set $f(\tau) = \tau$. The respective *relabelling* of an MTS Q is denoted by $Q[f]$ and obtained from Q by replacing Σ with Σ' and each action α of a transition with $f(\alpha)$.

Similarly, for an alphabet H, the *hiding* of H in Q, denoted by Q/H, is obtained from Q by replacing Σ with $\Sigma \setminus H$ and each action $a \in H$ of a transition with τ. ◊

The idea of parallel composition is that two systems synchronize on common (visible) actions and perform all other actions independently.

Definition 4 (MTS parallel composition). The *MTS parallel composition* of two MTS Q_1 and Q_2 is defined as the MTS $Q_1 \| Q_2 = (S_1 \times S_2, \Sigma_1 \cup \Sigma_2, \dashrightarrow, \longrightarrow, (s_1^0, s_2^0))$ with

$$\begin{aligned} \longrightarrow = \{((s_1, s_2), \alpha, (s_1', s_2)) \mid s_1 \xrightarrow{\alpha}_1 s_1' \ \wedge \ \alpha \notin \Sigma_2\} \\ \cup \{((s_1, s_2), \alpha, (s_1, s_2')) \mid s_2 \xrightarrow{\alpha}_2 s_2' \ \wedge \ \alpha \notin \Sigma_1\} \\ \cup \{((s_1, s_2), a, (s_1', s_2')) \mid s_1 \xrightarrow{a}_1 s_1' \ \wedge \ s_2 \xrightarrow{a}_2 s_2' \ \wedge \ a \in \Sigma_1 \cap \Sigma_2\} \end{aligned}$$

and $\dashrightarrow$ is defined analogously. ◊

Note that two equally labelled must-transitions synchronize to a must-transition, and the same for may-transitions. In effect, a must- and a may-transition synchronize to a may-transition, because the must-transition has an *underlying* may-transition. Finally, we define a variant of parallel composition where the synchronized actions are hidden.

Definition 5 (MTS parallel composition with hiding). The *parallel composition with hiding* of MTS Q_1 and Q_2 is the MTS $Q_1 \Uparrow Q_2 = (Q_1 \| Q_2)/H$ with $H = \Sigma_1 \cap \Sigma_2$. ◊

In [9], there is a parametric precongruence result for modal refinement (in a version for MTS without an initial state), which can be instantiated to obtain the following result. The details have been worked out in [17].

Theorem 6. *For relabelling, hiding, parallel composition and parallel composition with hiding, $\sqsubseteq_{MTS}$ is a precongruence, i.e.: for MTS Q_1, Q_2 and R with $Q_1 \sqsubseteq_{MTS} Q_2$, a relabelling function f for Q_1 (and thus for Q_2), and an alphabet H we have:*

$$\begin{gathered} Q_1[f] \sqsubseteq_{MTS} Q_2[f], \quad Q_1/H \sqsubseteq_{MTS} Q_2/H, \\ Q_1 \| R \sqsubseteq_{MTS} Q_2 \| R, \quad Q_1 \Uparrow R \sqsubseteq_{MTS} Q_2 \Uparrow R \end{gathered}$$

Since (Petri net) liveness is an issue in the related literature, we define a corresponding property on MTS in such a way that it is preserved under refinement. An action a is action live in an MTS if it surely remains possible whatever happens. Formally:

Definition 7 (action live). For an MTS Q, $a \in \Sigma$ is *action live* in Q if, for each reachable state s, $s \stackrel{wa}{\Longrightarrow} s'$ for some word $w \in \Sigma^\tau$.

Proposition 8. *For MTS Q_1 and Q_2 with $Q_1 \sqsubseteq_{MTS} Q_2$, $a \in \Sigma_1$ is action live in Q_2 implies a is action live in Q_1.*

Proof. The assumptions imply that there is a suitable MTS-relation $\mathfrak{R}$. A reachable state s_1 of Q_1 is reached by a sequence of may-transitions. Each of these is matched by a small path in Q_2 according to $\mathfrak{R}$; stringed together, these paths reach some s_2 with $(s_1, s_2) \in \mathfrak{R}$. By assumption for a, there is some w and s_2' with $s_2 \stackrel{wa}{\Longrightarrow} s_2'$. In turn, the respective must-transitions are matched in Q_1, implying $s_1 \stackrel{wa}{\Longrightarrow} s_1'$. □

For implementations, action liveness directly corresponds to Petri net liveness. In Definition 7, s is reached by may-transitions and a is performed along a sequence of must-transitions. To see that this is the right choice of modalities, think of a variant where only states s reachable by must-transitions are considered. If Q consists of states s^0 and s with $s^0 \xrightarrow{a} s^0$ and $s^0 \stackrel{\tau}{\dashrightarrow} s$, then a would be action live in Q, but not in a refinement having the τ-transition as a must. Vice versa, think of a variant where it suffices that a is performed along a sequence of may-transitions. If Q consists of state s^0 with $s^0 \stackrel{a}{\dashrightarrow} s^0$, then a would be action live in Q, but not in a refinement having no transition.

2.2 Modalities for Petri Nets

Also for Petri nets, one can distinguish between must- and may-transitions. Additionally, one can label transitions with actions, which form an interface for synchronous communication (MPN). Alternatively, one can distinguish specific input and output places, and these form an interface for asynchronous communication (MON). Our focus lies on the latter, but we need also MPN for our approach, and we even need a combination for the envisaged coarsest precongruence result. For generality, we start from this combination. Note that all transitions are may-transitions, their set is denoted by T as usual. We also treat infinite nets, but observe the assumption in the paragraph after the following definition.

Definition 9 (ℓMON). A *labelled modal open net* (ℓMON) is a tuple

$$N = (P, I, O, \Sigma, T, T^{\Box}, W, m^0, l)$$

where P and T are disjoint sets of *places* and *(may-)transitions*, and $T^{\Box} \subseteq T$ is the set of *must-transitions*; $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is the set of weighted *arcs*; m^0 is the *initial marking*, where a *marking* is a mapping $m : P \to \mathbb{N}$.

Furthermore, $I \subseteq P$ and $O \subseteq P$ are disjoint sets of *input* and *output places*, which are empty under the initial marking. Finally, Σ is an alphabet disjoint from I and O, and $l : T \to \Sigma^{\tau}$ is the *labelling*; τ-labels are omitted in figures.

A *modal open net* (MON) is an ℓMON where Σ is empty, cf. Fig. 1; we will often omit Σ and l, which maps all transitions to τ. A *modal Petri net* (MPN) is an ℓMON where I and O are empty and often omitted. ◊

We call $F = \{(x,y) \mid W(x,y) \neq 0\}$ the *flow relation* of N. For an $x \in P \dot\cup T$, we call the sets ${}^\bullet x = \{y \mid (y,x) \in F\}$ the *preset* and $x^\bullet = \{y \mid (x,y) \in F\}$ the *postset* of x. At some stage, we will need that transitions have finite presets, so we *assume* this throughout.

The behaviour of an ℓMON N is given by the occurrence rule. A transition $t \in T$ is *enabled* at a marking m, if $\forall p \in {}^\bullet t : W(p,t) \leq m(p)$. When t is enabled at m, it can *occur* or *fire*, changing the marking to m' with $m'(p) = m(p) - W(p,t) + W(t,p)$; we write $m \overset{t}{\dashrightarrow} m'$, or $m \xrightarrow{t} m'$ if t is a must-transition. Furthermore, the same notation is used for transition labels, i.e. we also write $m \overset{l(t)}{\dashrightarrow} m'$ or $m \xrightarrow{l(t)} m'$.

The latter notations in fact define the may- and must-transitions of an MTS *associated* to N: its alphabet is Σ, m^0 the initial state, and the reachable markings are the states. With this view, the other MTS notations like $\overset{w}{\dashrightarrow}$ and $\overset{w}{\Longrightarrow}$ for words carry over to ℓMON. Whenever $m \overset{w}{\dashrightarrow} m'$ or $m \overset{w}{=\!\Rightarrow} m'$, there exists an *underlying* transition sequence, a *firing sequence* leading from m to m'.

2.3 MPN: Refinement and Operators

First, we will concentrate on MPN. With the concept of an associated MTS, MPN refinement can be defined according to Definition 2, i.e. the MPN-relation below is just an MTS-relation between the associated MTS:

Definition 10 (MPN refinement). For MPN N_1 and N_2 over the same alphabet, we say that N_1 is a *refinement* of N_2, written $N_1 \sqsubseteq_{MPN} N_2$, if there is an *MPN-relation* $\mathfrak{R}$ between the reachable markings of N_1 and N_2 with $(m_1^0, m_2^0) \in \mathfrak{R}$ such that for every $(m_1, m_2) \in \mathfrak{R}$:

- $m_2 \xrightarrow{\alpha} m_2' \;\Rightarrow\; m_1 \overset{\widehat{\alpha}}{\Longrightarrow} m_1' \wedge (m_1', m_2') \in \mathfrak{R}$ and
- $m_1 \overset{\alpha}{\dashrightarrow} m_1' \;\Rightarrow\; m_2 \overset{\widehat{\alpha}}{=\!\Rightarrow} m_2' \wedge (m_1', m_2') \in \mathfrak{R}$. ◊

In some cases, we might use MPN-relations that include unreachable markings. This can make arguments easier, e.g. we do not have to prove reachability. Strictly, we would have to remove all pairs containing an unreachable marking.

Next, we define the operations for MTS also for MPN. The essential point for parallel composition is that, for a common label a, each a-labelled transition in the first and each a-labelled transition in the second MPN are merged to a new transition, which inherits both presets and both postsets. This implies the lemma after the definition. Note that we identify isomorphic structures; hence, we can e.g. assume place sets to be disjoint in this definition:

Definition 11 (MPN operators). Let N_1 and N_2 be MPN, where w.l.o.g. the place sets are disjoint. Then, we define their *parallel composition* to be

$$N_1 \| N_2 = (P, \Sigma, T, T^\square, W, m^0, l)$$

where P and Σ are the componentwise unions, i.e. $P = P_1 \cup P_2$ etc.

– $T = \{(t_1, \tau) \mid t_1 \in T_1 \ \wedge \ l_1(t_1) \notin \Sigma_2\} \cup \{(\tau, t_2) \mid t_2 \in T_2 \ \wedge \ l_2(t_2) \notin \Sigma_1\}$
$\cup \{(t_1, t_2) \mid t_1 \in T_1 \ \wedge \ t_2 \in T_2 \ \wedge \ l_1(t_1) = l_2(t_2) \in \Sigma_1 \cap \Sigma_2\}$,

– $T^{\Box}$ is defined analogously,

– $\forall p \in P, \ (t_1, t_2) \in T : \quad W(p, (t_1, t_2)) = \begin{cases} W_1(p, t_1) \text{ if } p \in P_1 \ \wedge \ t_1 \in T_1 \\ W_2(p, t_2) \text{ if } p \in P_2 \ \wedge \ t_2 \in T_2 \\ 0 \qquad\quad \text{otherwise} \end{cases}$
$W((t_1, t_2), p)$ is defined analogously,

– $\forall p \in P : m^0(p) = \begin{cases} m_1^0(p) \text{ if } p \in P_1 \\ m_2^0(p) \text{ if } p \in P_2, \end{cases}$

– $\forall (t_1, t_2) \in T : l(t_1, t_2) = \begin{cases} l_1(t_1) \text{ if } t_1 \in T_1 \\ l_2(t_2) \text{ if } t_2 \in T_2. \end{cases}$

With this, we define relabelling, hiding and parallel composition with hiding word by word as in Definitions 3 and 5. ◊

Note that in the last item above, in case of a merged transition, $l_1(t_1)$ and $l_2(t_2)$ coincide. For the next lemma, note that markings of $N_1 \| N_2$ can be written (m_1, m_2), where m_1 is a marking of N_1 and m_2 one of N_2.

Lemma 12. *Let N_1 and N_2 be two MPN. If t_1 and t_2 are a-labelled transitions of N_1 and N_2 resp., then $(m_1, m_2) \xrightarrow{(t_1,t_2)} (m_1', m_2')$ if and only if $m_1 \xrightarrow{t_1} m_1'$ and $m_2 \xrightarrow{t_2} m_2'$. If t_1 is a transition of N_1 with $l_1(t_1) \notin \Sigma_2$, then $(m_1, m_2) \xrightarrow{(t_1,\tau)} (m_1', m_2)$ if and only if $m_1 \xrightarrow{t_1} m_1'$, and analogously for N_2. The same statements hold for must-transitions.*

This lemma implies that the MTS associated to $N_1 \| N_2$ is the parallel composition of the two MTS associated to $N_1 \| N_2$. Similar statements hold for the other three operators defined above. Hence, we obtain the following corollary to Theorem 6.

Corollary 13. *W.r.t. the above operators for MPN, $\sqsubseteq_{MPN}$ is a precongruence.*

Observe that, in the same way, the notion of action liveness and its preservation under refinement carry over to MPN and $\sqsubseteq_{MPN}$. We close with a technical operation and lemma, which will be important in the next section. The operation contracts special τ-must-transitions by merging the only place in the preset with the only place in the postset. This is illustrated in Fig. 2.

Definition 14 (τ-contraction). Let N be an MPN and A a set of τ-labelled must-transitions with the following properties:

– for each $t_i \in A$: ${}^\bullet t_i = \{p_i\}$, $t_i^\bullet = \{p_i'\}$ and $W(p_i, t) = W(t, p_i') = 1$; furthermore, ${}^\bullet p_i' = p_i^\bullet = \{t_i\}$ and $m^0(p_i) = m^0(p_i') = 0$;

– all these places are different.

Then, the τ-*contraction* $N[A]$ is obtained from N by removing the transitions $t_i \in A$ and the associated places p'_i, and changing the values $W(p_i, t)$ for the remaining transitions t from 0 to $W(p'_i, t)$. ◊

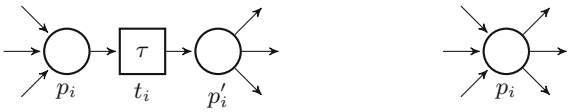

Fig. 2. Transformation from MPN N to MPN $N[A]$

Lemma 15. *Let N be an MPN and $A \subseteq T_N^{\Box}$ as in Definition 14, then $N \sqsubseteq_{MPN} N[A]$ and $N[A] \sqsubseteq_{MPN} N$.*

Proof. Let m and m' be markings of N and $N[A]$, resp., that are identical on the common places except that, for each of the p_i, $m'(p_i) = m(p_i) + m(p'_i)$; then we denote m' by $[m]$. Now consider the relation $\mathfrak{R} = \{(m, [m]) \mid m \text{ reachable in } N\}$. This relation proves the first claim, and its reverse proves the second claim.

First, consider some $m \xrightarrow{t_i} m'$ with $t_i \in A$. This can be matched by firing no transition: $(m', [m]) \in \mathfrak{R}$ since t_i does not change $m(p_i) + m(p'_i)$, i.e. $[m'] = [m]$. It remains to deal with each transition t of $N[A]$, and we restrict ourselves to must-transitions, since the case of may-transitions is similar. So second, consider $m \xrightarrow{t} m'$ in N. From each $p'_i \in {}^\bullet t$, t removes $W(p'_i, t)$ tokens, which are present in and removed from p_i under $[m]$; for the other places, t removes the same number of tokens in both nets, and it adds the same number of tokens to each place in both nets.

Third, consider $[m] \xrightarrow{t} [m']$ in $N[A]$. Here, some tokens on some $p'_i \in {}^\bullet t$ might be missing in N. This is remedied by first firing invisibly some t_i: for each of the (finitely many!) $p'_i \in {}^\bullet t$, we fire the invisible must-transitions t_i until p_i is empty. Now t removes the same number of tokens from $p'_i \in {}^\bullet t$ in N as it removes from p_i in $N[A]$, and it removes the same number of tokens in both nets from each other place; then, it adds the same number of tokens in both nets to each place. □

3 Asynchronous Communication

While the definition of composing nets according to asynchronous communication, i.e. by merging places, should be pretty clear, the question is how to define a refinement framework that deals with the interface places in a suitable way. The idea of [19,21] is to make visible how an environment interacts via these places. The environment observes that it puts a token onto an input place, but not when the token is taken, and vice versa for output places. Thus, for each input (output) place a, we add an arc from (to) a new a-labelled transition and compare the resulting MPNs with $\sqsubseteq_{MPN}$. In the following definition, we assume that a^- and a^+ are fresh in the sense that they are not in $P \cup T$.

Definition 16 (ℓMON wrapper). The ℓMON *wrapper* of an ℓMON N is the MPN $wrap(N)$ (also denoted here by N_w). N_w is obtained from N by renaming $a \in I$ ($a \in O$) to the fresh a^- (a^+) – defining P_w; these inherit the arcs and initial marking from a – defining m_w^0. We set $\Sigma_w = \Sigma \dot\cup I \dot\cup O$ and add a-labelled transitions a to T and $T^\Box$ for all $a \in I \cup O$ – defining also T_w, $T_w^\Box$ and l_w. The modified W is extended on the new pairs involving some new transition a by $W_w(a, a^-) = 1$ for $a \in I$, $W_w(a^+, a) = 1$ for $a \in O$ and 0 otherwise – defining W_w. See Fig. 3 for the example N_2 and $wrap(N_2)$.

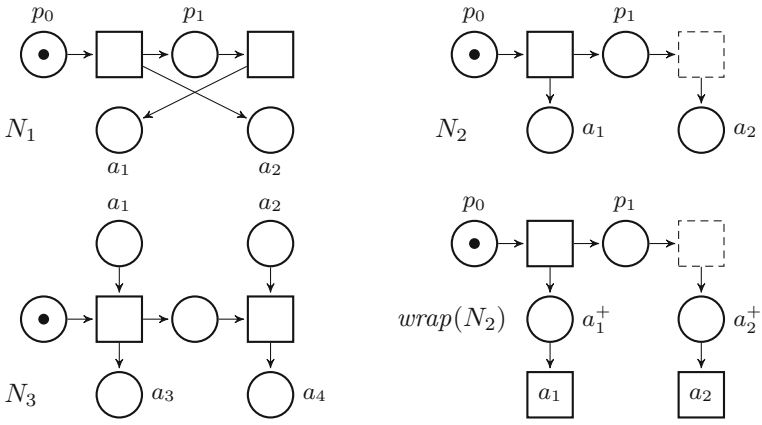

Fig. 3. Three MON N_1, N_2, N_3 and $wrap(N_2)$

Definition 17 (ℓMON refinement). Let N_1 and N_2 be two ℓMON with the same alphabet as well as input and output places. We say that N_1 is a *refinement* of N_2, written $N_1 \sqsubseteq_{\ell MON} N_2$, if $wrap(N_1) \sqsubseteq_{MPN} wrap(N_2)$. ◊

This definition extends Definition 10: since an MPN N is identical to $wrap(N)$, $\sqsubseteq_{\ell MON}$ and $\sqsubseteq_{MPN}$ coincide if applied to two MPN.

As a first example, we show that $N_1 \sqsubseteq_{\ell MON} N_2$ for the MON N_1 and N_2 in Fig. 3 by showing a suitable MPN-relation $\mathfrak{R}$. We write markings as a formal sum: e.g. if p_1 and a_1 have one token each, we write $p_1 + a_1$, and we write this also for a marking of $wrap(N_2)$ although the place has changed its name to a_1^+ there; 0 is the empty marking. With this, $\mathfrak{R} = \{(p_0, p_0), (p_1 + a_2, p_1 + a_1), (a_2, p_1), (p_1, a_1), (a_1 + a_2, a_1 + a_2), (a_1, a_1), (a_2, a_2), (0, 0)\}$.

An interesting pair is $(p_1 + a_2, p_1 + a_1)$: the only enabled must-transition on the specification side is a_1; although the token on a_1 is produced after the token on a_2 in N_1, this can be matched using the second τ-transition, which is a must transition. Thus, the pair (a_2, p_1) is reached. On the refinement side, a_2 is enabled, which can be matched with the second τ-transition in N_1; it is sufficient that this is a may-transition. Additionally, the two second τ-transitions match each other as required.

Here, the specification produces two messages in some order while the refinement produces them the other way round. We justify this intuitively after the definition of asynchronous composition.

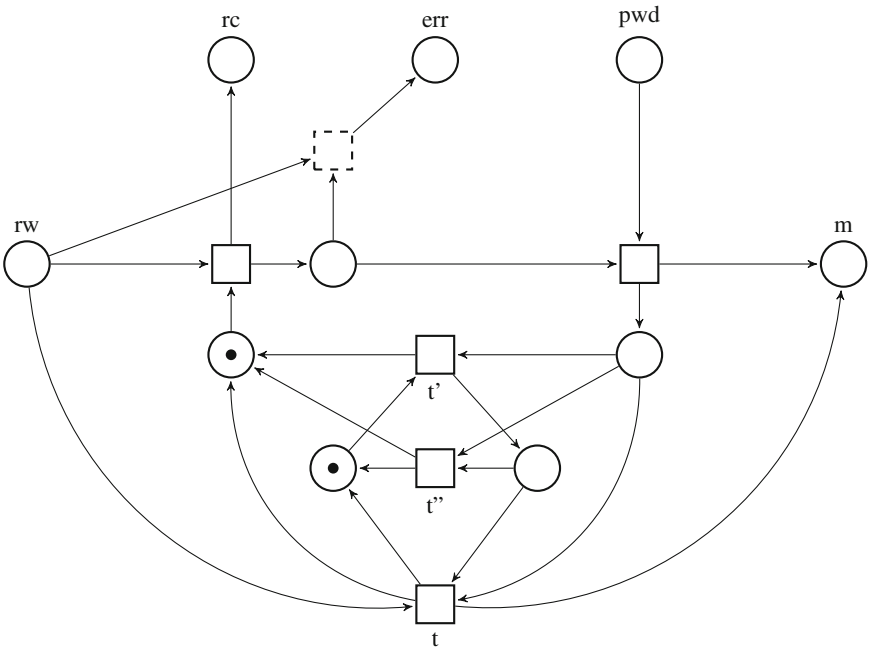

Fig. 4. Refinement ATM' of ATM

As another example, Fig. 4 shows a refinement ATM' of the lower MON ATM in Fig. 1. Here, the optional top right may-transition is omitted, possibly because we assume that the password provider in Fig. 1 is used and we expect no errors due to premature passwords. Furthermore, the optional shortcut t is now a must-transition. But there is a difference that makes it less obvious that ATM' really is a refinement of ATM: if the shortcut is used in ATM, it can only be used again the next but one time; in ATM', it can only be used one time later. Let us prove the refinement.

The two MON have the same places except for the lower two places of ATM'; obviously, these together will always have one token. For each reachable marking m of $wrap(ATM)$, we denote by $m+l$ ($m+r$) the same marking of $wrap(ATM')$ with an additional token on the left (right) additional place. The MPN-relation $\mathfrak{R}$ for $wrap(ATM)$ and $wrap(ATM')$ consists of all pairs $(m+l, m)$ and $(m+r, m)$ for such reachable m, obviously relating the initial markings.

The new visible must-transitions of $wrap(ATM)$ and the upper must-transitions of ATM have the same labels and effects in $wrap(ATM')$. Furthermore, $m \xrightarrow{\tau} m'$ due to t' in $wrap(ATM)$ if and only if $m + l \xrightarrow{\tau} m' + r$ due

to t' in $wrap(ATM')$ if and only if $m + r \xrightarrow{\tau} m' + l$ due to t'' in $wrap(ATM')$. Thus, all must- and their underlying may-transitions (except t) are matched appropriately.

The may-transition of $wrap(ATM')$ is matched by itself in $wrap(ATM)$. Finally, t can only fire under a marking $m + r$ in $wrap(ATM')$ resulting in $m + r \overset{\tau}{\dashrightarrow} m' + l$. This is matched in $wrap(ATM)$ by $m \overset{\tau}{\dashrightarrow} m'$ due to t.

We call some $a \in \Sigma \cup I \cup O$ *action live* in an ℓMON N if it is action live in $wrap(N)$. This is the case for each $a \in I$, whereas for $a \in O$ it means: N can always put another token onto a provided that sufficiently many tokens are provided on the input places. Obviously, action liveness is also preserved under $\sqsubseteq_{\ell MON}$.

We now come to the most important operator of this paper; it merges common interface places, modelling asynchronous communication.

Definition 18 (ℓMON asynchronous composition). Two ℓMON N_1 and N_2 are called *(async-)composable* whenever $(\Sigma_1 \cup I_1 \cup O_1) \cap (\Sigma_2 \cup I_2 \cup O_2) = (I_1 \cap O_1) \cup (I_2 \cap O_2) =: asc(N_1, N_2)$. We can further assume that the four place and transition sets are pairwise disjoint except for $asc(N_1, N_2)$.

The *(asynchronous) composition* of such ℓMON is the ℓMON $N_1 \oplus N_2 = (P, I, O, \Sigma, T, T^{\square}, W, m^0, l)$ where P, Σ, T and $T^{\square}$ are the componentwise unions. The interface places are $I = (I_1 \cup I_2) \setminus asc(N_1, N_2)$ and $O = (O_1 \cup O_2) \setminus asc(N_1, N_2)$. For $i = 1, 2$, marking m^0 coincides on $p \in P_i$ with m_i^0, and l coincides on $t \in T_i$ with l_i. Finally,

$$W(p,t) = \begin{cases} W_1(p,t) & \text{if } p \in P_1 \wedge t \in T_1 \\ W_2(p,t) & \text{if } p \in P_2 \wedge t \in T_2 \\ 0 & \text{otherwise} \end{cases} \quad \text{— } W(t,p) \text{ is defined analogously.} \qquad \Diamond$$

Composability ensures that $N_1 \oplus N_2$ is well-defined. In particular, it ensures that synchronous and asynchronous channels do not get confused. One could also think of a variant that combines parallel and asynchronous composition where the components also synchronize on common actions – actions we have forbidden here. We observe that the composition $\oplus$ is commutative and associative up to isomorphism for pairwise composable components. Note that for three ℓMON N_1, N_2 and N_3 with some $a \in I_1 \cap O_2 \cap I_3$, $(N_1 \oplus N_2) \oplus N_3$ and $N_1 \oplus (N_2 \oplus N_3)$ might be well-defined, but would have different behaviour in general: N_2 communicates on a with N_1 in one and with N_3 in the other composition. So also N_1 and N_3 have to be composable.

Let us reconsider the nets N_1 and N_2 in Fig. 3. As a potential argument against reordering messages, one might come up with N_3, which looks like it is sensitive to the order in which tokens arrive on a_1 and a_2. In $N_2 \oplus N_3$, the first token arrives on a_1, and a token on a_3 can be produced immediately; in $N_1 \oplus N_3$, the first token arrives on a_2. But using the second τ-transition of N_1, a token can be put onto a_3 before marking a_4, so N_3 cannot "see" the reordering. Thus, this reordering *should* be allowed in a refinement step. To prove that $\sqsubseteq_{\ell MON}$ is indeed a precongruence w.r.t. $\oplus$, we need another lemma.

Lemma 19. *Let N_1 and N_2 be two composable ℓMON and $A = \{(a,a) \mid a \in asc(N_1,N_2)\}$, then $(wrap(N_1) \Uparrow wrap(N_2))[A]$ and $wrap(N_1 \oplus N_2)$ are isomorphic.*

Proof. By composability, $asc(N_1,N_2)$ is the set of the common actions of $wrap(N_1)$ and $wrap(N_2)$. For $wrap(N_1) \Uparrow wrap(N_2)$, the unique a-labelled transition a in one net is merged with the unique a-labelled transition a in the other if $a \in asc(N_1,N_2)$, and then all these a-labels are hidden. Hence, A is a set as required in Definition 14 and $[A]$ merges a^+ and a^- into one place, which we may call a again; cf. Fig. 2. Now the only difference is that transitions are pairs, where always one component is τ. Removing these components results in $wrap(N_1 \oplus N_2)$. □

Theorem 20. *The refinement relation $\sqsubseteq_{\ell MON}$ is a precongruence for $\oplus$, i.e. for three ℓMON N_1, N_2 and N_3 where N_2 is composable with N_3 and $N_1 \sqsubseteq_{\ell MON} N_2$, also N_1 is composable with N_3 and $N_1 \oplus N_3 \sqsubseteq_{\ell MON} N_2 \oplus N_3$.*

Proof. Composability only depends on the interfaces, so the first claim is obvious. Let $A = \{(a,a) \mid a \in asc(N_1,N_3)\} = \{(a,a) \mid a \in asc(N_2,N_3)\}$.

By definition of $\sqsubseteq_{\ell MON}$ and the precongruence properties of $\sqsubseteq_{MPN}$, we have

– $wrap(N_1) \Uparrow wrap(N_3) \sqsubseteq_{MPN} wrap(N_2) \Uparrow wrap(N_3)$.

Now by Lemmas 19 and 15,

– $wrap(N_1 \oplus N_3) \sqsubseteq_{MPN} (wrap(N_1) \Uparrow wrap(N_3))[A]$
  $\sqsubseteq_{MPN} wrap(N_1) \Uparrow wrap(N_3)$ and
– $wrap(N_2) \Uparrow wrap(N_3) \sqsubseteq_{MPN} (wrap(N_2) \Uparrow wrap(N_3))[A]$
  $\sqsubseteq_{MPN} wrap(N_2 \oplus N_3)$.

Thus, $wrap(N_1 \oplus N_3) \sqsubseteq_{MPN} wrap(N_2 \oplus N_3)$ and $N_1 \oplus N_3 \sqsubseteq_{\ell MON} N_2 \oplus N_3$. □

It might seem that the *wrap*-based definition of our refinement relation is somewhat arbitrary; our next aim is to show its optimality by proving a coarsest-precongruence result. The starting point is that modal refinement is accepted for MTS, so its translation $\sqsubseteq_{MPN}$ to MPN is in a sense just right. Hence, the optimal refinement relation $\sqsubseteq$ on ℓMON should respect this: if $N_1 \sqsubseteq N_2$ for MPN N_1 and N_2, then also $N_1 \sqsubseteq_{MPN} N_2$. Furthermore, $\sqsubseteq$ should be a precongruence w.r.t. $\oplus$. To be optimal, it should allow all refinements consistent with these two requirements, so $\sqsubseteq$ should be the coarsest MPN-respecting precongruence w.r.t. $\oplus$; such a coarsest precongruence always exists.

In principle, this coarsest precongruence could be finer than $\sqsubseteq_{MPN}$ for MPN but – being $\sqsubseteq_{\ell MON}$ – it actually coincides with $\sqsubseteq_{MPN}$ for MPN, which is even more pleasing.

One could question why a precongruence for all ℓMON is needed. Our proof below also supports another argument with the same starting point to answer this. In this argument, we call an ℓMON N_o an *observer* of a MON N if it has the same interface places as N but with input and output interchanged; thus,

$N \oplus N_o$ is an MPN. Hence, N_o interacts with N on the complete asynchronous interface of the latter, and it can make its observations visible on its synchronous interface. Now one could alternatively aim for some $\sqsubseteq$ on MON that is the coarsest precongruence w.r.t. $\oplus$ such that $N \sqsubseteq N'$ implies, for all observers N_o of N (i.e. also of N'), that $N \oplus N_o \sqsubseteq_{MPN} N' \oplus N_o$. Again, this $\sqsubseteq$ is $\sqsubseteq_{\ell MON}$ (restricted to MON). In fact, for all MON N and N', $N \sqsubseteq_{\ell MON} N'$ if and only if $N \oplus N_o \sqsubseteq_{MPN} N' \oplus N_o$ for all observers N_o of N, as we show more generally in the next proposition.

Definition 21. A relation $\sqsubseteq$ on ℓMON is called *MPN-respecting* if it implies $\sqsubseteq_{MPN}$ on MPN. An ℓMON N_o is an *observer* of an ℓMON N, if N_o and N are composable, $I_o = O$ and $O_o = I$. ◊

Proposition 22. *Let N and N' be ℓMON with the same alphabet as well as input and output places. Then $N \sqsubseteq_{\ell MON} N'$ if and only if $N \oplus N_o \sqsubseteq_{MPN} N' \oplus N_o$ for all observers N_o of N.*

Proof. For the "if"-direction, we construct a specific N_o: it has, for each $a \in I \cup O$, an empty place a and an a'-labelled transition, where a' is a fresh action, i.e. $a' \notin \Sigma \cup I \cup O$. (These fresh actions are needed, since an interface place is not allowed to be an action as well.) The only arcs have weight one and connect each $a \in I_o$ to the a'-labelled transition and, for $a \in O_o$, the a'-labelled transition to the place a. Further, let f be the relabelling that maps each a' to a and is the identity on Σ.

Now, $N \oplus N_o$ is isomorphic to $wrap(N)$ except that it has labels a' instead of a and, so, $(N \oplus N_o)[f]$ is isomorphic to $wrap(N)$. Thus, $N \oplus N_o \sqsubseteq_{MPN} N' \oplus N_o$ implies $(N \oplus N_o)[f] \sqsubseteq_{MPN} (N' \oplus N_o)[f]$, which implies $N \sqsubseteq_{\ell MON} N'$.

The "only if"-direction follows from Theorem 20 and the observation after Definition 17.

Theorem 23. *Relation $\sqsubseteq_{\ell MON}$ is the coarsest MPN-respecting precongruence w.r.t. $\oplus$ on ℓMON.*

Proof. Let $\sqsubseteq$ be the coarsest MPN-respecting precongruence w.r.t. $\oplus$ on ℓMON. Due to Theorem 20 and the observation after Definition 17, $\sqsubseteq_{\ell MON}$ is an MPN-respecting precongruence as well. Thus, it is contained in $\sqsubseteq$ by the definition of the latter.

This definition also gives us that $N \sqsubseteq N'$ implies $N \oplus N_o \sqsubseteq N' \oplus N_o$ and $N \oplus N_o \sqsubseteq_{MPN} N' \oplus N_o$ for all observers N_o of N. The latter implies $N \sqsubseteq_{\ell MON} N'$ by Proposition 22, showing that $\sqsubseteq$ is contained in $\sqsubseteq_{\ell MON}$ as well. □

We close with a quick look at the operators that we have only defined for MPN so far. The following definition extends Definition 11 to ℓMON; in particular, par-composability holds automatically for MPN.

Definition 24 (further ℓMON operators). Two ℓMON N_1 and N_2 are called *par-composable* whenever $(\Sigma_1 \cup I_1 \cup O_1) \cap (\Sigma_2 \cup I_2 \cup O_2) = \Sigma_1 \cap \Sigma_2$. We can further

assume that the place sets are disjoint. Then, we define their *parallel composition* $N_1 \| N_2$ as in Definition 11, letting also I and O be the componentwise unions.

For an ℓMON N, a *relabelling function* f is defined as in Definition 3 except that, additionally, we require that Σ' and $I \cup O$ be disjoint. With this, *relabelling* $N[f]$ and *hiding* N/H are defined word by word as in Definition 3. Similarly, the *parallel composition with hiding* $N_1 \Uparrow N_2$ of two par-composable ℓMON N_1 and N_2 is defined as before as $(N_1 \| N_2)/H$ with $H = \Sigma_1 \cap \Sigma_2$. $\Diamond$

Also the operations $\|$ and $\Uparrow$ are commutative and associative up to isomorphism for pairwise par-composable components.

Theorem 25. *The relation $\sqsubseteq_{\ell MON}$ is a precongruence for $\|$ and $\Uparrow$ on ℓMON, i.e. for three ℓMON N_1, N_2 and N_3 where N_2 is par-composable with N_3 and $N_1 \sqsubseteq_{\ell MON} N_2$, also N_1 is par-composable with N_3, $N_1 \| N_3 \sqsubseteq_{\ell MON} N_2 \| N_3$ and $N_1 \Uparrow N_3 \sqsubseteq_{\ell MON} N_2 \Uparrow N_3$. The relation is also a precongruence for relabelling and hiding.*

Proof. For parallel composition, observe that $wrap(N_i) \| wrap(N_3)$ and $wrap(N_i \| N_3)$ are isomorphic for $i = 1, 2$, since *wrap* adds the same transitions to the same places for both systems, and these new transitions are also not synchronized in the first system. By definition of $\sqsubseteq_{\ell MON}$, we have $wrap(N_1) \sqsubseteq_{MPN} wrap(N_2)$, which implies $wrap(N_1) \| wrap(N_3) \sqsubseteq_{MPN} wrap(N_2) \| wrap(N_3)$ by Corollary 13. The above observation gives $wrap(N_1 \| N_3) \sqsubseteq_{MPN} wrap(N_2 \| N_3)$ and we are done.

For $N_1 \sqsubseteq_{\ell MON} N_2$ and a suitable relabelling function f, we extend f to f_{IO}, which additionally is the identity on $I_1 \cup O_1$; f_{IO} is a relabelling function for each $wrap(N_i)$. Now we only have to observe that $wrap(N_i)[f_{IO}])$ and $wrap(N_i[f])$ are isomorphic. With this, we are done as above, using again the definition of $\sqsubseteq_{\ell MON}$ and Corollary 13.

The case of hiding is easier, and the case of $\Uparrow$ is implied. $\square$

4 Modal Asynchronous I/O-Petri Nets (MAP)

For comparison, we have a closer look at MAP, which are MPN where the visible actions are subdivided into input, output and internal actions [8]. An input action a indicates that an a-labelled transition takes a token from the (only implicit) place a, and analogously for an output. For composition, a common action a must always be an input of one and an output of the other component. A new place a is created and connected to a-labelled transitions as explained above; it represents an internal channel of the overall system. The label a is changed to *internal* actions $a^{\triangleright}$ on the output and ${}^{\triangleright}a$ on the input side.

The main issue in [8] is to decide the property *message consuming* (and a variation thereof): a net is message consuming w.r.t. internal channel a if, whenever there is a token on a, it is possible to perform a must-${}^{\triangleright}a$, possibly preceded by output, internal or hidden must-transitions. This is regarded as a quality criterion for communication since no message in a channel will necessarily

be ignored. Message consuming is preserved under composition and refinement, and to achieve this, internal actions must be visible.

Message consuming can be too strict: possibly, a message on a can only be processed sensibly if another message on b is received first. Also, a message consumption is certainly not so relevant if it has no effect for the environment. So we will not pursue this issue here. But note that the main idea in [19] is similar in spirit: there, the aim is to construct systems that only stop when a final marking (from a predefined set) is reached where all channel places are empty. Thus, the system will not stop while a message is pending. In contrast to the MAP approach, this property is not checked for the components; the aim is only to achieve it in the final system where it really is essential.

Since it is argued in [8] that, for stepwise refinement, the internal transitions should be hidden in the end, we will do so immediately in our comparison. To avoid a partitioning of Σ, we present MAP as MPN where the visible actions have the form $a^\triangleright$ or ${}^\triangleright a$ and, for no a, we have $a^\triangleright$ *and* ${}^\triangleright a$ in Σ. The refinement is simply $\sqsubseteq_{MPN}$. MAP are *composable* if their alphabets are disjoint. In the composition $\oplus_{HH}$, whenever some $a^\triangleright$ is in one and ${}^\triangleright a$ in the other alphabet, a new place a is created together with an arc from each $a^\triangleright$-labelled transition and an arc to each ${}^\triangleright a$-labelled transition as sketched in Fig. 5 on the left; the respective transitions are hidden.

Essentially, we could produce the same net by adding a place a and the resp. connections to each of the two MAP first and then apply $\oplus$. The first part gives us a function that embeds MAP into MON.

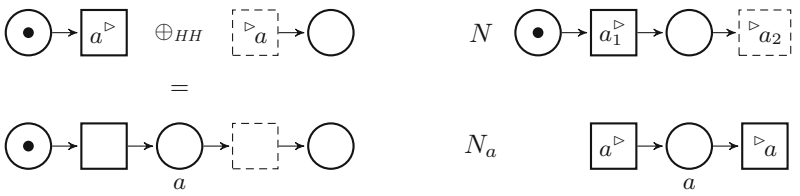

Fig. 5. Composition $\oplus_{HH}$, MAP N and the MPN N_a

Definition 26. The function *map2mon* maps each MAP N to a MON by adding for each $a^\triangleright$ and ${}^\triangleright a$ in Σ a new empty place a together with a weight-1 arc from each $a^\triangleright$-labelled transition or an arc to each ${}^\triangleright a$-labelled transition resp. All actions are hidden.

For a symbol a, N_a denotes an MPN (not a MAP!) as shown in Fig. 5. For a set A, we denote the disjoint union of the N_a with $a \in A$ by N_A. ◊

For the MAP N in Fig. 5, *map2mon*(N) is the MON N_2 in Fig. 3. Clearly, any *map2mon*(N) is a MON with the restriction (violated by the MON in Fig. 1) that each transition is only connected to at most one interface place, and then with an arc of weight one. From each such MON, we can shear off the interface places

and reconstruct the respective transition labels for a corresponding MAP. The following theorem states that MAP is a proper sub-setting of our MON-setting with a stricter refinement.

Theorem 27. *Function map2mon embeds MAP into MON in the sense that it is injective but not surjective and, for all composable MAP N_1 and N_2, map2mon(N_1) and map2mon(N_2) are composable and*

$$map2mon(N_1 \oplus_{HH} N_2) = map2mon(N_1) \oplus map2mon(N_2) \ .$$

If we have $N_1 \sqsubseteq_{MPN} N_2$ instead, then map2mon(N_1) $\sqsubseteq_{\ell MON}$ map2mon(N_2), but not vice versa.

Proof. The first sentence should be clear. For the refinement, let $A = \{a \mid a^{\triangleright}$ or ${}^{\triangleright}a$ in $\Sigma\}$ and let f rename $a^{\triangleright}$ (!) to a if ${}^{\triangleright}a \in \Sigma$ and ${}^{\triangleright}a$ (!) to a if $a^{\triangleright} \in \Sigma$. Now, $(N_i \Uparrow N_A)[f]$ is isomorphic to $wrap(map2mon(N_i))$. Since $(N_1 \Uparrow N_A)[f] \sqsubseteq_{MPN} (N_2 \Uparrow N_A)[f]$ by the MPN-precongruence results, we are done.

Finally, we prove that the implication is strict. Consider the MAP N in Fig. 5 with $map2mon(N) = N_2$ in Fig. 3, and the similar MAP N' with $map2mon(N') = N_1$ in Fig. 3. We have $N_1 \sqsubseteq_{\ell MON} N_2$, but $N \sqsubseteq_{MPN} N'$ fails due to the reordering. □

In [8], it is shown that $\sqsubseteq_{MPN}$ on MAP is a precongruence for $\oplus_{HH}$ with a (not so difficult) proof that goes into the details of the definition of $\sqsubseteq_{MPN}$. One can also prove this from general precongruence results on MPN. Let N_1, N_2 and N_3 be MAP such that $N_1 \sqsubseteq_{MPN} N_2$ and N_1 and N_3 are composable. Let A be the set of those a where ${}^{\triangleright}a$ is in one of Σ_1 and Σ_3 while $a^{\triangleright}$ is in the other. Then, $N_i \Uparrow N_A \Uparrow N_3$ is isomorphic to $N_i \oplus_{HH} N_3$ and, with $N_1 \Uparrow N_A \Uparrow N_3 \sqsubseteq_{MPN} N_2 \Uparrow N_A \Uparrow N_3$, we are done.

5 Bounded Modal Open Nets

To argue that $N_1 \sqsubseteq_{\ell MON} N_2$ for MON N_1 and N_2, we have to exhibit an MPN-relation for $wrap(N_1)$ and $wrap(N_2)$. The problem is that the latter usually have infinitely many reachable markings, since arbitrarily many tokens can be put on each input place. One solution to this problem is to work with finite nets and to require that the final system (a closed MON as defined below) is *b-bounded* for some fixed bound b, i.e. that all reachable markings are b-bounded, assigning at most b tokens to each place. We sketch below how to modify MPN-relations for a setting where more than b tokens on a place are considered to be an error.

It can also be helpful to observe that a $wrap(N)$ is a special MPN, where each visible action a appears just once, and on a must-transition. From the position of such a transition, we can read off whether a is an input or an output action and whether the incident place (still denoted by a below) was an input or output place originally without having specific components in the MPN-tuple. We give here a first observation only, calling an MPN a *special MPN (sMPN)* if it is $wrap(N)$ for some MON N.

Proposition 28. *Let $\mathfrak{R}$ be an MPN-relation for sMPN N_1 and N_2, and $(m_1, m_2) \in \mathfrak{R}$. Writing $m + i$ for a marking m with an additional token on input place i, also $\mathfrak{R} \cup \{(m_1 + i, m_2 + i)\}$ is an MPN-relation for N_1 and N_2.*

Proof. We check the two conditions for $(m_1 + i, m_2 + i)$.

— Let $m_2 + i \xrightarrow{\alpha} m_2''$. By $(m_1, m_2) \in \mathfrak{R}$ and $m_2 \xrightarrow{i} m_2 + i$, there is some $(m_1', m_2 + i) \in \mathfrak{R}$ with $m_1 \stackrel{i}{\Longrightarrow} m_1'$; we can assume that the underlying firing sequence starts with $m_1 \xrightarrow{i} m_1 + i$, since the i-transition does not remove any token; thus, $m_1 + i \Longrightarrow m_1'$. Furthermore, $m_1' \stackrel{\alpha}{\Longrightarrow} m_1''$ with $(m_1'', m_2'') \in \mathfrak{R}$. Hence, $m_1 + i \stackrel{\alpha}{\Longrightarrow} m_1''$ matches $m_2 + i \xrightarrow{\alpha} m_2''$.

— Let $m_1 + i \stackrel{\alpha}{\dashrightarrow} m_1''$. By $(m_1, m_2) \in \mathfrak{R}$ and $m_1 \stackrel{i}{\dashrightarrow} m_1 + i$, there is some $(m_1 + i, m_2') \in \mathfrak{R}$ with $m_2 \stackrel{i}{=\!\Rightarrow} m_2'$; we can again assume that the underlying firing sequence starts with $m_2 \stackrel{i}{\dashrightarrow} m_2 + i$, so that $m_2 + i =\!\Rightarrow m_2'$. Furthermore, $m_2' \stackrel{\alpha}{=\!\Rightarrow} m_2''$ with $(m_1'', m_2'') \in \mathfrak{R}$. Hence, $m_2 + i \stackrel{\alpha}{=\!\Rightarrow} m_2''$ matches $m_1 + i \stackrel{\alpha}{\dashrightarrow} m_1''$. □

This observation shows that $m_1 \xrightarrow{i} m_1 + i$ can always be matched by $m_2 \xrightarrow{i} m_2 + i$ and vice versa; no other pair than $(m_1 + i, m_2 + i)$ is needed for this in $\mathfrak{R}$. This can help to prove or disprove $N_1 \sqsubseteq_{MPN} N_2$.

Often, it is desirable that systems are finite state and channels have a finite capacity. The final systems in such a setting can be modelled by finite b-bounded Petri nets; for the rest of this section, we fix some arbitrary positive bound b.

Definition 29. A MON is *closed* if it has no input or output places. A marking m of a MON or an sMPN that is not b-bounded is called an *error*; then, a marking m' with $m' =\!\Rightarrow m$ is called *illegal*. ◊

A closed MON describes a final system, which usually arises as the composition of a system with the final user. We consider a setting where such a closed MON is required to be b-bounded. Note that a closed MON N coincides with $wrap(N)$.

If such a MON is built with a system component N', a marking of $wrap(N')$ that is not b-bounded is an *error*; it cannot occur in the final system and subsequent behaviour is irrelevant. In fact, this already holds for an illegal marking m', since nothing can prevent $wrap(N')$ to move autonomously from m' to an error. Note that the occurrence of a transition t can only lead from a legal to an illegal marking if t is an input.

Interface automata (IA) [5] form a similar setting (with synchronous communication), where an "unexpected" input leads to an error. While IA are a kind of LTS, there is quite some literature on combinations with modalities, see [4] for an advanced approach called modal interface automata (MIA). Similarly to transferring refinement and precongruence results from MTS to MPN, one can transfer these with some care from MIA to sMPN. The refinement definition looks as follows; note that any behaviour is better than an error, so an illegal marking does not have to be matched.

Definition 30 (sMPN-b-refinement). For sMPN N_1 and N_2 with the same input and output actions, we say that N_1 is an *sMPN-b-refinement* of N_2, written $N_1 \sqsubseteq^b_{sMPN} N_2$, if there is an *sMPN-$b$-relation* $\mathfrak{R}$ between the reachable markings of N_1 and N_2 with $(m_1^0, m_2^0) \in \mathfrak{R}$ such that for every $(m_1, m_2) \in \mathfrak{R}$ where m_2 is legal:

- m_1 is legal,
- $m_2 \xrightarrow{\alpha} m_2' \;\Rightarrow\; m_1 \stackrel{\widehat{\alpha}}{\Longrightarrow} m_1' \wedge (m_1', m_2') \in \mathfrak{R}$,
- $m_1 \stackrel{\alpha}{\dashrightarrow} m_1' \;\Rightarrow\; m_2 \stackrel{\widehat{\alpha}}{=\!\Rightarrow} m_2' \wedge (m_1', m_2') \in \mathfrak{R}$. ◊

For closed MON N_1 and N_2, $wrap(N_1) \sqsubseteq^b_{sMPN} wrap(N_2)$ simply means that m_1^0 must be legal if m_2^0 is. Intuitively, this means: if the system specification composed with the user is b-bounded, then the refinement composed with the user is b-bounded as well.

Some details are simpler for sMPN than for MIA. Here, m' is illegal if $m' =\!\Rightarrow m$ for some error m. For MIA, also outputs must be considered for the transition sequence; we can ignore these here, since output transitions only remove tokens. Furthermore, for the matching of transitions as in Definition 30, inputs and outputs are treated differently for MIA: in case of an input, the matching transition sequence must *start* with the respective input. Here, this does not matter; if the only visible transition in a firing sequence is an input, we can just as well move it to the front since it does not remove tokens.

Additionally, MIA have so-called disjunctive must-transitions [4] for defining conjunction on MIA. It is not at all clear to us how a conjunction for Petri nets ("real" Petri nets with concurrency) could look like. Compared to a setting without conjunction, disjunctive must-transitions make [4] unnecessarily difficult to read. Therefore, we plan to work out a self-contained presentation of the b-bounded setting. There, it will be worthwhile to explicitly accompany each sMPN by a modified reachability graph, where – as in MIA – all illegal markings are merged into a special error state.

As a final remark, we point out that our b-bounded setting is *optimistic* like IA and MIA. An sMPN might have behaviour that leads to an error. As long as an error cannot be reached autonomously (i.e. the initial marking is illegal), there is an environment such that the composition is error-free; e.g. the environment may simply not provide any inputs. In fact, if the respective MON has some input place, reachable errors are unavoidable. A *pessimistic* approach as in [3] forbids components where errors are reachable, it cannot be applied here.

6 Conclusion

In [6,8], Petri nets were augmented with may- and must-modalities and modal refinement for stepwise design, and they were used for modelling asynchronous communication via merging implicit interface places (MAP). We have here applied a much older framework for nets with interface places [21], see also [19], and developed an according refinement relation for nets with modalities and

explicit interface places (MON). We have justified this relation with a coarsest-precongruence result. Our studies were carried out in a larger setting with modal nets having interface places for asynchronous as well as action-labelled transitions for synchronous communication.

Details of the MAP-approach are related to checking so-called *message consumption*, a property that holds if, intuitively speaking, each message can eventually be received, i.e. removed from the channel. For stepwise refinement, it is more appropriate to abstract from these details, as also suggested in [6,8]. With this abstraction, it turned out that MAP is a subsetting of MON with a stricter refinement relation. With an example, we have shown that some reordering of messages leads to a rejection as a refinement in the MAP-approach, although it is intuitively acceptable for asynchronous communication (and in the MON-approach).

To show that one MON refines another, a suitable alternating simulation has to be exhibited. These simulations have special properties (compared to the general alternating simulations used for MAP), which could help to find one or prove that none exists. We have given one such property here and will look into this issue in the future.

Often, it is desirable that the components are finite-state and channels have a finite capacity. We have given a rough sketch how this can be integrated into the MON-approach and plan to work this out in detail. Furthermore, also motivated by the idea of message consumption, we think about integrating final markings as in [19] such that a system can only stop when all channels are empty.

References

1. Baldan, P., Corradini, A., Ehrig, H., Heckel, R.: Compositional modeling of reactive systems using open nets. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 502–518. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44685-0_34
2. Baldan, P., Corradini, A., Ehrig, H., Heckel, R., König, B.: Bisimilarity and behaviour-preserving reconfigurations of open Petri Nets. Log. Methods Comput. Sci. **4**(4) (2008). https://doi.org/10.2168/LMCS-4(4:3)2008
3. Bauer, S.S., Mayer, P., Schroeder, A., Hennicker, R.: On weak modal compatibility, refinement, and the MIO workbench. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 175–189. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12002-2_15
4. Bujtor, F., Fendrich, S., Lüttgen, G., Vogler, W.: Nondeterministic modal interfaces. Theoret. Comput. Sci. **642**, 24–53 (2016)
5. de Alfaro, L., Henzinger, T.A.: Interface-based design. In: Broy, M., Grünbauer, J., Harel, D., Hoare, T. (eds.) Engineering Theories of Software Intensive Systems. NSS, vol. 195, pp. 83–104. Springer, Dordrecht (2005). https://doi.org/10.1007/1-4020-3532-2_3
6. Elhog-Benzina, D., Haddad, S., Hennicker, R.: Refinement and asynchronous composition of modal petri nets. In: Jensen, K., Donatelli, S., Kleijn, J. (eds.) Transactions on Petri Nets and Other Models of Concurrency V. LNCS, vol. 6900, pp. 96–120. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29072-5_4

7. Glabbeek, R.J.: The linear time — branching time spectrum II. In: Best, E. (ed.) CONCUR 1993. LNCS, vol. 715, pp. 66–81. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57208-2_6
8. Haddad, S., Hennicker, R., Møller, M.H.: Specification of asynchronous component systems with modal I/O-petri nets. In: Abadi, M., Lluch Lafuente, A. (eds.) TGC 2013. LNCS, vol. 8358, pp. 219–234. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05119-2_13
9. Hüttel, H., Larsen, K.G.: The use of static constructs in a model process logic. In: Meyer, A.R., Taitslin, M.A. (eds.) Logic at Botik 1989. LNCS, vol. 363, pp. 163–180. Springer, Heidelberg (1989). https://doi.org/10.1007/3-540-51237-3_14
10. Kindler, E.: A compositional partial order semantics for Petri net components. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 235–252. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63139-9_39
11. Larsen, K.G., Thomsen, B.: A modal process logic. In: Logic in Computer Science 1988, pp. 203–210. IEEE (1988)
12. Massuthe, P., Reisig, W., Schmidt, K.: An operating guideline approach to the SOA. Ann. Math. Comput. Teleinformatics **1**, 35–43 (2005)
13. Milner, R.: Communication and Concurrency. Prentice-Hall, Inc., Upper Saddle River (1989)
14. Pomello, L.: Some equivalence notions for concurrent systems. An overview. In: Rozenberg, G. (ed.) APN 1985. LNCS, vol. 222, pp. 381–400. Springer, Heidelberg (1986). https://doi.org/10.1007/BFb0016222
15. Raclet, J.B.: Residual for component specifications. Electr. Notes Theor. Comput. Sci. **215**, 93–110 (2008)
16. Reisig, W.: Deterministic buffer synchronization of sequential processes. Acta Inf. **18**, 117–134 (1982)
17. Schneider, V.: A better semantics for asynchronously communicating Petri nets. M.Sc. Thesis, Universität Augsburg (2017)
18. Souissi, Y.: On liveness preservation by composition of nets via a set of places. In: Rozenberg, G. (ed.) ICATPN 1990. LNCS, vol. 524, pp. 277–295. Springer, Heidelberg (1991). https://doi.org/10.1007/BFb0019979
19. Stahl, C., Vogler, W.: A trace-based service semantics guaranteeing deadlock freedom. Acta Inf. **49**, 69–103 (2012)
20. Vogler, W.: Behaviour preserving refinements of Petri nets. In: Tinhofer, G., Schmidt, G. (eds.) WG 1986. LNCS, vol. 246, pp. 82–93. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-17218-1_51
21. Vogler, W.: Modular Construction and Partial Order Semantics of Petri Nets. LNCS, vol. 625. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55767-9