Trace- and Failure-Based Semantics for Responsiveness

Walter Vogler[1] and Christian Stahl[2] and Richard Müller[2,3]

[1] Institut für Informatik, Universität Augsburg, Germany
vogler@informatik.uni-augsburg.de
[2] Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, The Netherlands
c.stahl@tue.nl
[3] Institut für Informatik, Humboldt-Universität zu Berlin, Germany
richard.mueller@informatik.hu-berlin.de

Abstract. We study open systems modeled as Petri nets with an interface for asynchronous (i.e., buffered) communication with other open systems. As a minimal requirement for successful communication, we investigate *responsiveness*, which guarantees that an open system and its environment always have the possibility to communicate. We investigate responsiveness with and without final states and also their respective bounded variants, where the number of pending messages never exceeds a previously known bound. Responsiveness *accordance* describes when one open system can be safely replaced by another open system. We present a trace-based characterization for each accordance variant. As none of the relations turns out to be compositional (i.e., it is no precongruence), we characterize the *coarsest compositional relation* (i.e., the coarsest precongruence) that is contained in each relation, using a variation of should testing. For the two unbounded variants, the precongruences are not decidable, but for the two bounded variants we show *decidability*.

Keywords: Petri nets, Asynchronous communication, Compositionality, Precongruence, Should testing preorder

1 Introduction

Today's software systems are complex distributed systems that are composed of less complex *open systems*. In this paper, we focus on open systems that have a well-defined interface and communicate with each other via *asynchronous message passing*. Service-oriented systems like Web-service applications [27] and systems based on wireless network technologies like wireless sensor networks [2], medical systems, transportation systems, or online gaming are examples of such distributed systems. During system evolution, often one open system is replaced by another one—for example, when new features have been implemented or

bugs have been fixed. This requires a *refinement* notion, which should respect *compositionality*.

In this paper, we model an open system as a Petri net extended with an interface for asynchronous communication. As a *minimal requirement* for successful communication, *responsiveness* demands that an open system and its environment (called a *controller*) always have the possibility to communicate. An open system is in responsiveness *accordance* with another one, if it can replace the latter as part of a closed system without affecting this property. The related property of deadlock freedom can be satisfied by one component that works internally forever. Responsiveness has gained interest because it additionally ensures the possibility to communicate, which is crucial in the setting of interacting open systems. An example is Microsoft's asynchronous event driven programming language P, which is used to implement device drivers [11]. P programs can be checked for a stricter variant of responsiveness, which additionally requires that no message in any channel is ignored forever.

We present a *trace-based characterization* of responsiveness accordance. The semantics consists of a set of completed traces and responsiveness accordance is characterized by trace inclusion. Usually, controller-based preorders like ours are precongruences and, thus, a compositional refinement notion on open systems; as this is not the case for our responsiveness accordance, we characterize the *coarsest precongruence* that is contained in this preorder. Interestingly, we obtain Vogler's $\mathcal{F}^+$-semantics [33] (which has been later introduced as impossible futures [36]), and the corresponding precongruence is the should testing preorder [25,6,29]. Such a characterization is vital, because the definition of a coarsest precongruence considers arbitrary parallel environments and is, therefore, hard to check in concrete cases. Also, a declarative characterization furthers understanding and can, for example, help to decide the precongruence.

In addition, we study responsiveness in the presence of *final states*. Again, we characterize the preorder based on traces. To distinguish final and nonfinal states we collect successfully and unsuccessfully completed traces. As also this preorder is not a precongruence, we characterize the coarsest precongruence as a variant of should testing. More precisely, we have to add an additional set to the $\mathcal{F}^+$-semantics collecting all traces that do not lead to a final state. The precongruence can be defined in line with the should testing preorder.

In unpublished work, we showed that neither the preorders nor the precongruences are decidable. These results are not in the scope of this paper, as the proof technique applied is very different from those we use here. This motivates us to investigate beside final and nonfinal states a second dimension of responsiveness, *boundedness*, resulting in two additional variants of responsiveness. For these two variants, we require the composition of two open systems to be finite-state and, in particular, that the number of pending messages never exceeds a previously known bound. This is practically relevant: Distributed systems operate on a middleware with buffers that are of bounded size. The actual buffer size can be the result of a static analysis of the underlying middleware or of the communication behavior of an open system, or simply be chosen sufficiently large.

We give a trace-based characterization for each bounded variant of responsiveness accordance, thereby adapting and combining results from the unbounded variants and work on traces that cannot be used reliably by any controller [21]. Due to the latter traces, accordant systems may violate language inclusion. Giving an answer to an open question, we show again that none of the accordance variants turns out to be a precongruence. So we characterize for each variant its coarsest precongruence that is contained in the preorder. To this end, we add information about bound violations to the precongruence of the respective unbounded variant. In fact, as we require a user to define the upper bound b of pending messages, we obtain a *family* of preorders and precongruences, each parameterized by b. Based on our characterization, we prove the two coarsest precongruences in the bounded case to be *decidable*: for the first precongruence, the problem can be reduced to deciding should testing [29]; for the second, we refine the proof in [29] by further details.

The goal of this article is to contribute to a general theory on open systems in the presence of an asynchronous unqueued (i.e., buffered) communication scheme. Although we present only the theory, open systems specified in industrial languages such as WS-BPEL or BPMN can be translated into our formal model and then be analyzed [19].

Our contribution can be summarized as follows:

- We give a *trace-based characterization for the four variants of responsiveness accordance.*
- For each variant of accordance, we characterize the *coarsest precongruence* that is contained in the respective preorder as variants of the should testing preorder.
- For the two bounded variants, we show decidability of the precongruences.

This article extends previous work of the same authors [35,34]. In [35], we presented an extended abstract of unbounded responsiveness accordance and the respective variant in the presence of final states. In this article, we present these two variants including all proofs. The work in [34], is an extended abstract of the bounded variant of responsiveness accordance without final states and presents only a sketch of the main results of the respective variant in the presence of final states. In this article, we present all the results of the bounded variants including all proofs.

After some background in Sect. 2, Sect. 3 introduces the basic variant, responsiveness (i.e., with possibly unbounded message buffers and without final states), characterizes the respective accordance relation semantically and presents a characterization of the coarsest precongruence that is contained in this relation. The presence of final states is investigated in Sect. 4. Section 5 prepares for dealing with boundedness and presents a precongruence for boundedness without considering responsiveness. Next, Sect. 6 introduces bounded responsiveness, characterizes the respective accordance relation semantically, presents a characterization of the coarsest precongruence that is contained in this relation, and proves its decidability. Section 7 characterizes bounded responsiveness in the

presence of final states. We discuss related work in Sect. 8 and close with a conclusion in Sect. 9.

2 Preliminaries

This section provides the basic notions, such as Petri nets, open nets for modeling open systems, and open net environments for describing the semantics of open nets.

For two sets A and B, let $A \uplus B$ denote the disjoint union; writing $A \uplus B$ implies that A and B are implicitly assumed to be disjoint. Let $\mathbb{N}$ denote the natural numbers including 0.

2.1 Petri nets

As a basic model, we use place/transition Petri nets extended with a set of final markings and transition labels.

Definition 2.1 (net). A *net* $N = (P, T, F, m_N, \Omega)$ consists of

- a set P of *places*,
- a set T of *transitions* such that P and T are disjoint,
- a *flow relation* $F \subseteq (P \times T) \uplus (T \times P)$,
- an *initial marking* m_N, where a marking is a mapping $m : P \to \mathbb{N}$, and
- a set Ω of *final markings*.

Usually, we are interested in finite nets—that is, nets with finite sets P and T—but for some results (e.g., Theorems 3.17 and 4.17), we also make use of infinite nets.

Introducing a net N also implicitly introduces its components P, T, F, m_N, Ω; the same applies to nets N', N_1, etc. and their components $P', T', F', m_{N'}, \Omega'$, and $P_1, T_1, F_1, m_{N_1}, \Omega_1$, respectively—and it also applies to other structures later on.

Definition 2.2 (labeled net). A *labeled net* $N = (P, T, F, m_N, \Omega, \Sigma_{in}, \Sigma_{out}, l)$ is a net (P, T, F, m_N, Ω) together with an *alphabet* $\Sigma = \Sigma_{in} \uplus \Sigma_{out}$ of disjoint *input actions* Σ_{in} and *output actions* Σ_{out} and a *labeling function* $l : T \to \Sigma \uplus \{\tau\}$, where τ represents an invisible, internal action. Two labeled nets are *interface-equivalent* if they have the same sets of input and output actions.

Graphically, a circle represents a place, a box represents a transition, and the directed arcs between places and transitions represent the flow relation. In the case of a labeled net, a transition label is depicted inside a transition with bold font to distinguish it from the transition's identity. A marking is a distribution of tokens over the places. Graphically, a black dot represents a token.

Let $x \in P \uplus T$ be a node of a net N. As usual, ${}^\bullet x = \{y \mid (y, x) \in F\}$ denotes the *preset* of x and $x^\bullet = \{y \mid (x, y) \in F\}$ the *postset* of x. We interpret presets and postsets as multisets when used in operations also involving multisets.

A marking is a multiset over the set P of places; for example, $[p_1, 2p_2]$ denotes a marking m with $m(p_1) = 1$, $m(p_2) = 2$, and $m(p) = 0$ for $p \in P \setminus \{p_1, p_2\}$. We define $+$ and $-$ for the sum and the difference of two markings and $=, <, >, \leq, \geq$ for comparison of markings in the standard way. We canonically extend the notion of a marking of N to supersets $Q \supseteq P$ of places; that is, for a mapping $m : P \to \mathbb{N}$, we extend m to the marking $m : Q \to \mathbb{N}$ such that for all $p \in Q \setminus P$, $m(p) = 0$. Conversely, a marking can be restricted to a subset $Q \subseteq P$ of the places of N. For a mapping $m : P \to \mathbb{N}$, the *restriction* of m to the places in Q is denoted by $m|_Q : Q \to \mathbb{N}$.

Let Σ_1, Σ_2 be alphabets. For a word $w \in \Sigma_1^*$ and $\Sigma_2 \subseteq \Sigma_1$, $w|_{\Sigma_2}$ denotes the projection of w to the subalphabet Σ_2. With $v \sqsubseteq w$ we denote that v is a *prefix* of w. We write $|w|$ for the length of w, and $|w|_x$ denotes how many times $x \in \Sigma$ occurs in word w. As usual, ε denotes the empty word.

The *behavior* of a net N relies on the marking of N and changing the marking by the firing of transitions of N. A transition $t \in T$ is *enabled* at a marking m, denoted by $m \xrightarrow{t}$, if for all $p \in {}^\bullet t$, $m(p) > 0$. If t is enabled at m, it can *fire*, thereby changing the current marking m to a marking $m' = m - {}^\bullet t + t^\bullet$. The firing of t is denoted by $m \xrightarrow{t} m'$; that is, t is enabled at m and firing it results in m'.

The behavior of N can be extended to sequences: $m_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{k-1}} m_k$ is a *run* of N if for all $0 < i < k$, $m_i \xrightarrow{t_i} m_{i+1}$. A marking m' is *reachable from* a marking m if there exists a (possibly empty) run $m_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{k-1}} m_k$ with $m = m_1$ and $m' = m_k$; for $v = t_1 \ldots t_{k-1}$, we also write $m_1 \xrightarrow{v} m_k$. A marking m' is *reachable* if it is reachable from the initial marking. The set M_N represents the set of all reachable markings of N.

In the case of labeled nets, we lift runs to traces: If $m_1 \xrightarrow{v} m_k$ and w is obtained from v by replacing each transition with its label and removing all τ labels, we write $m_1 \stackrel{w}{\Longrightarrow} m_k$ and refer to w as a *trace* whenever $m_1 = m_N$. The *language* $L(N)$ of a labeled net N is the set of all traces of N. The *reachability graph* $RG(N)$ *of* N has the reachable markings M_N as its nodes and a t-labeled edge from m to m' whenever $m \xrightarrow{t} m'$ in N. In the case of a labeled net, each edge label t is replaced with $l(t)$.

Finally, we introduce boundedness of nets. A marking m of a net N is *b-bounded* for a bound $b \in \mathbb{N}$, if $m(p) \leq b$ for all $p \in P$. The net N is *b-bounded* if every reachable marking is b-bounded; it is *bounded*, if it is b-bounded for some $b \in \mathbb{N}$. Throughout the paper, b denotes a bound—a positive natural number.

2.2 Open nets

Like Lohmann et al. [18] and Stahl et al. [30], we model open systems as *open nets* [33,18], thereby restricting ourselves to the communication protocol of an open system. An open net extends a net by an interface. An interface consists of two disjoint sets of input and output places corresponding to asynchronous input and output channels. In the model, we abstract from data and represent each message by a token on the respective interface place. In the initial marking

and the final markings, interface places are not marked. An input place has an empty preset, and an output place has an empty postset. We consider only open nets that have either at least one input and one output place or no input and output places; open nets with just input or just output places cannot really take part in a responsive communication.

Definition 2.3 (open net). An *open net* N is a tuple $(P, T, F, m_N, I, O, \Omega)$ such that

- $(P \uplus I \uplus O, T, F, m_N, \Omega)$ is a net;
- for all $p \in I \uplus O$, $m_N(p) = 0$ and for all $m \in \Omega$, $m(p) = 0$;
- the set I of *input places* satisfies for all $p \in I$, ${}^\bullet p = \emptyset$;
- the set O of *output places* satisfies for all $p \in O$, $p^\bullet = \emptyset$; and
- set $I = \emptyset$ if and only if set $O = \emptyset$.

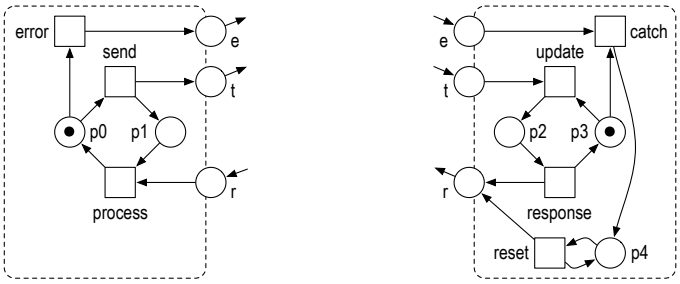
If $I = O = \emptyset$, then N is a *closed net*. The net $\mathit{inner}(N)$ results from removing the interface places and their incident arcs from N. Two open nets are *interface equivalent* if they have the same sets of input and output places.

Graphically, we represent an open net like a net with a dashed frame around it. An interface place p is positioned on the frame; an additional arrow indicates whether p is an input or an output place.

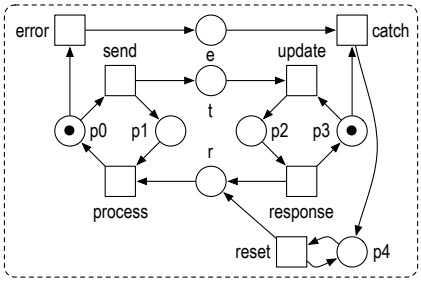
For the composition of open nets, we assume that the sets of transitions are pairwise disjoint and that no internal place of an open net is a place of any other open net. In contrast, the interfaces intentionally overlap. We require that all communication is *bilateral* and *directed*; that is, every shared place p has only one open net that sends into p and one open net that receives from p. In addition, we require that either all interface places are shared or there is at least one input and one output place which are not shared. We refer to open nets that fulfill these conditions as *composable*. We compose two composable open nets N_1 and N_2 by merging shared interface places and turning these places into internal places. The definition of *composable* thereby guarantees that an open net composition is again an open net (possibly a closed net).

Definition 2.4 (open net composition). Two open nets N_1 and N_2 are *composable* if $(P_1 \uplus T_1 \uplus I_1 \uplus O_1) \cap (P_2 \uplus T_2 \uplus I_2 \uplus O_2) = (I_1 \cap O_2) \uplus (I_2 \cap O_1)$, and $(I_1 \uplus I_2) \setminus (O_1 \uplus O_2)$ and $(O_1 \uplus O_2) \setminus (I_1 \uplus I_2)$ are both either empty or nonempty. The *composition* of two composable open nets N_1 and N_2 is the open net $N_1 \oplus N_2 = (P, T, F, m_N, I, O, \Omega)$, where

- $P = P_1 \uplus P_2 \uplus (I_1 \cap O_2) \uplus (I_2 \cap O_1)$,
- $T = T_1 \uplus T_2$,
- $F = F_1 \uplus F_2$,
- $m_N = m_{N_1} + m_{N_2}$,
- $I = (I_1 \uplus I_2) \setminus (O_1 \uplus O_2)$,
- $O = (O_1 \uplus O_2) \setminus (I_1 \uplus I_2)$, and
- $\Omega = \{m_1 + m_2 \mid m_1 \in \Omega_1, m_2 \in \Omega_2\}$.



(a) Open net S (b) Open net C



(c) Open net $S \oplus C$

Fig. 1: Open nets modeling an unreliable time server, a client, and their composition. In addition to the models, we have $\Omega_S = \{[]\}$ and $\Omega_C = \Omega_{S \oplus C} = \{[p_3]\}$.

Example 2.1. Figure 1 shows three open systems, each modeled as an open net. The open net S models an unreliable time server that sends its timing information (output place t) to some client and processes its responses (input place r). Anytime before sending the timing information, an error may happen (output place e) and the server shuts down (and final marking $[]$ can be reached). The open net C models a client. It repeatedly updates its system time by the timing information sent by the server (input place t) and responds with a response packet (output place r). If the client receives an error message from the server (input place e), it continuously tries to reset the time server (output place r). The open nets S and C are composable. Their composition $S \oplus C$ is a closed net, which is depicted in Fig. 1c. The place r in $S \oplus C$ is unbounded; thus, the composition is unbounded, too.

2.3 Environments

To give an open net N a trace-based semantics, we consider its environment $env(N)$, which we define similarly to Vogler [33]. The net $env(N)$ can be constructed from N by adding to each interface place $p \in I$ ($p \in O$) a p-labeled transition p in $env(N)$ and renaming the place p to p^i (p^o). The net $env(N)$

shows the possible behaviour of an environment of N—that is, which inputs it can send to N and which outputs it can receive from N. It is just a tool to define our characterizations and prove our results. But intuitively, one can understand the construction as translating the asynchronous interface p of N into a buffered synchronous interface (with unbounded buffers p^i or p^o) described by the transition labels of $env(N)$.

Definition 2.5 (open net environment). The *environment of an open net* N is the labeled net $env(N) = (P \uplus P^I \uplus P^O, T \uplus I \uplus O, F', m_N, \Omega, I, O, l')$, where

- $P^I = \{p^i \mid p \in I\}$,
- $P^O = \{p^o \mid p \in O\}$,
- $F' = \quad ((P \uplus T) \times (T \uplus P)) \cap F$
  $\uplus\ \{(p^i, t) \mid p \in I, t \in T, (p,t) \in F\}$
  $\uplus\ \{(t, p^o) \mid p \in O, t \in T, (t,p) \in F\}$
  $\uplus\ \{(p^o, p) \mid p \in O\}$
  $\uplus\ \{(p, p^i) \mid p \in I\}$, and
- $l'(t) = \begin{cases} \tau, & t \in T \\ t, & t \in I \uplus O. \end{cases}$

Convention: Throughout the paper, each trace set and semantics for labeled nets is implicitly extended to any open net N via $env(N)$—for example, the *language* of N is defined as $L(N) = L(env(N))$.

To compose environments of composable open nets in particular and labeled nets in general, we define a parallel composition operator $\|$ where, for each action a that the components have in common, each a-labeled transition of one component is synchronized with each a-labeled transition of the other. In addition, we define a second parallel composition operator $\Uparrow$. This operator works as operator $\|$ and, in addition, hides all common actions—that is, changes the respective labels to τ. Hiding and $\|$ are defined as in [33].

Definition 2.6 (parallel composition and hiding). Two labeled nets N_1 and N_2 are *composable* if $P_1 \cap P_2 = \Sigma_{in1} \cap \Sigma_{in2} = \Sigma_{out1} \cap \Sigma_{out2} = \emptyset$. The *parallel composition* of two composable labeled nets is the labeled net $N_1 \| N_2 = (P, T, F, m_N, \Omega, \Sigma_{in}, \Sigma_{out}, l)$, where

- $P = P_1 \uplus P_2$,
- $T = \quad \{(t_1, t_2) \in T_1 \times T_2 \mid l_1(t_1) = l_2(t_2) \neq \tau\}$
  $\uplus\ \{(t_1, \tau) \in T_1 \times \{\tau\} \mid l_1(t_1) \notin \Sigma_2\}$
  $\uplus\ \{(\tau, t_2) \in \{\tau\} \times T_2 \mid l_2(t_2) \notin \Sigma_1\}$,
- $F = \quad \{(p, (t_1, t_2)) \in P \times T \mid (p, t_1) \in F_1 \vee (p, t_2) \in F_2\}$
  $\uplus\ \{((t_1, t_2), p) \in T \times P \mid (t_1, p) \in F_1 \vee (t_2, p) \in F_2\}$,
- $m_N = m_{N_1} + m_{N_2}$,
- $\Omega = \{m_1 + m_2 \mid m_1 \in \Omega_1, m_2 \in \Omega_2\}$,
- $\Sigma_{in} = (\Sigma_{in1} \uplus \Sigma_{in2}) \setminus (\Sigma_{out1} \uplus \Sigma_{out2})$,
- $\Sigma_{out} = \Sigma_{out1} \uplus \Sigma_{out2}$, and

– $l(t_1, t_2) = \begin{cases} l_1(t_1), & t_1 \in T_1 \\ l_2(t_2), & \text{otherwise.} \end{cases}$

For a labeled net N and a set $A \subseteq \Sigma$, we obtain N/A from N by *hiding* all actions of A, meaning we replace the respective labels in A with τ. The *parallel composition with hiding* is the labeled net $N_1 \Uparrow N_2 = (N_1 \| N_2)/(\Sigma_1 \cap \Sigma_2)$.

Example 2.2. Figure 2 shows the environments of the open nets S and C from Fig. 1, their parallel composition $env(S) \| env(C)$, and their parallel composition with hiding $env(S) \Uparrow env(C)$.

To describe the behavior of compositions, we define parallel compositions of words and languages; operator $\|$ synchronizes common actions, operator $\Uparrow$ also hides them. Observe that in $env(N_1) \Uparrow env(N_2)$ only common transitions are merged; operator $\|$ is needed to relate the respective transition sequences.

Definition 2.7. Let Σ_1, Σ_2 be alphabets and $\Sigma = (\Sigma_1 \cup \Sigma_2) \setminus (\Sigma_1 \cap \Sigma_2)$. Let $w_1 \in \Sigma_1^*$ and $w_2 \in \Sigma_2^*$ be words, and let $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ be languages. We define[4]

– $w_1 \| w_2 = \{w \in (\Sigma_1 \cup \Sigma_2)^* \mid w|_{\Sigma_1} = w_1, w|_{\Sigma_2} = w_2\}$,
– $w_1 \Uparrow w_2 = \{w|_\Sigma \mid w \in w_1 \| w_2\}$,
– $L_1 \| L_2 = \bigcup\{w_1 \| w_2 \mid w_1 \in L_1, w_2 \in L_2\}$, and
– $L_1 \Uparrow L_2 = \bigcup\{w_1 \Uparrow w_2 \mid w_1 \in L_1, w_2 \in L_2\}$.

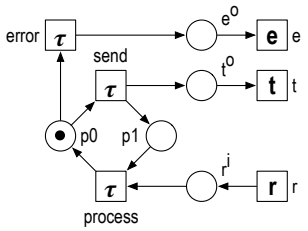
The next proposition recalls [33, Theorem 3.1.7(4)] and relates a trace of a composed labeled net to traces of its components.

Proposition 2.8 ([33]). *For two markings m_1 and m_2 of composable labeled nets N_1 and N_2, respectively, we have $m_1 + m_2 \stackrel{w}{\Longrightarrow} m_1' + m_2'$ in $N_1 \| N_2$ iff there exist $w_1 \in \Sigma_{N_1}^*$, $w_2 \in \Sigma_{N_2}^*$ such that $w \in w_1 \| w_2$, $m_1 \stackrel{w_1}{\Longrightarrow} m_1'$ in N_1, and $m_2 \stackrel{w_2}{\Longrightarrow} m_2'$ in N_2.*

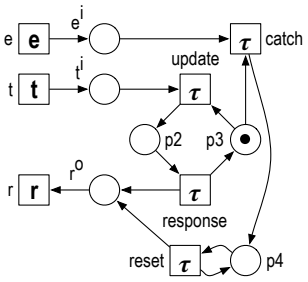
For the remainder of this section, we fix two composable open nets N_1 and N_2, and we put $C = env(N_1 \oplus N_2)$, $E = env(N_1) \| env(N_2)$, and $\overline{E} = env(N_1) \Uparrow env(N_2)$. The labeled nets $\overline{E}$ and E differ only in their labelings; C and $\overline{E}$ (E) have the same places, except for places $p \in (I_1 \cap O_2) \uplus (I_2 \cap O_1)$ in C and the corresponding places p^i, p^o in $\overline{E}$ (E). We study the relation between reachable markings of different compositions of N_1 and N_2. To this end, we define *agreement* between markings.

Definition 2.9 (agreement). A marking m of C and a marking $\overline{m}$ of $\overline{E}$ (of E) *agree* if they coincide on the common places and if for each $p \in (I_1 \cap O_2) \uplus (I_2 \cap O_1)$, $\overline{m}(p^o) + \overline{m}(p^i) = m(p)$. They *strongly agree* if, additionally, $\overline{m}(p^o) = 0$.

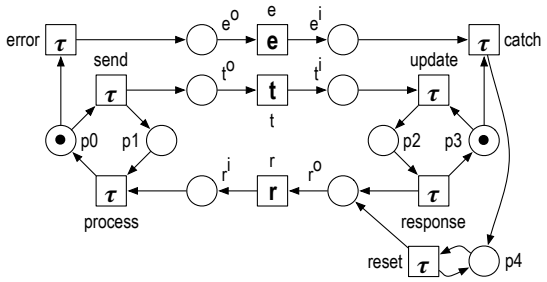
[4] To simplify the notation, we do not add the alphabets of w_1 and w_2 to operators $\|$ and $\Uparrow$; the alphabets will be always clear from the context.



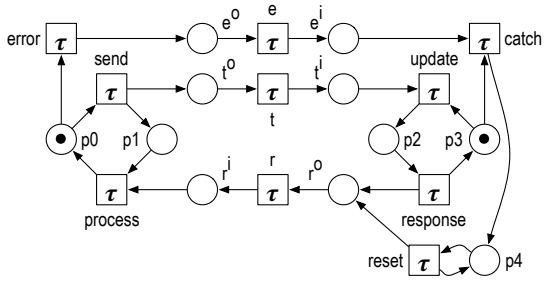
(a) Labeled net $env(S)$



(b) Labeled net $env(C)$



(c) Labeled net $env(S) \| env(C)$



(d) Labeled net $env(S) \Uparrow env(C)$

Fig. 2: The environments of the time server and the client and their parallel compositions. In addition to the models, we have $\Omega_{env(S)} = \{[\,]\}$, $\Omega_{env(C)} = \Omega_{env(S)\|env(C)} = \Omega_{env(S)\Uparrow env(C)} = \{[p_3]\}$.

The next proposition recalls [32, Lemma 15] and relates the firing of a transition of C and $\overline{E}$ (E).

Proposition 2.10 ([32]). *Let the markings m of C and $\overline{m}$ of $\overline{E}$ agree, and let $t \in T_C$. Then*

1. *If $\overline{m} \xrightarrow{t} \overline{m}'$, then $m \xrightarrow{t} m'$ such that m' and $\overline{m}'$ agree.*

2. *If m and $\overline{m}$ strongly agree, all additional transitions of $\overline{E}$ are disabled at $\overline{m}$, and further $m \xrightarrow{t} m'$ implies $\overline{m} \xrightarrow{t} \overline{m}'$ such that m' and $\overline{m}'$ agree.*

As $\overline{E}$ and E differ only in their labelings, (1)–(2) also hold for E in place of $\overline{E}$.

The next proposition recalls [32, Lemma 16(1),(2)] and [32, Lemma 15(4)]. It proves facts about sequences of transitions and relates final states.

Proposition 2.11 ([32]). *Let m be a marking of C and $\overline{m}$ be one of $\overline{E}$. Then*

1. *If m and $\overline{m}$ strongly agree and $m \xrightarrow{v} m'$ in C, then it is possible to insert transitions from $(I_1 \cap O_2) \uplus (I_2 \cap O_1)$ of $\overline{E}$ into v such that for the resulting v': $\overline{m} \xrightarrow{v'} \overline{m}'$ in $\overline{E}$ and also m' and $\overline{m}'$ strongly agree.*
2. *If m and $\overline{m}$ agree and $\overline{m} \xrightarrow{v'} \overline{m}'$ in $\overline{E}$, then it is possible to delete transitions from $(I_1 \cap O_2) \uplus (I_2 \cap O_1)$ of $\overline{E}$ in v' such that for the resulting v: $m \xrightarrow{v} m'$ in C and also m' and $\overline{m}'$ agree.*
3. *$m \in \Omega_C$ iff $\overline{m} \in \Omega_{\overline{E}}$ iff for $i = 1, 2$: $\overline{m}|_{P_{env(N_i)}} \in \Omega_{env(N_i)}$.*

As $\overline{E}$ and E differ only in their labelings, (1)-(3) also hold for E in place of $\overline{E}$.

Agreement between markings of C and $\overline{E}$ is a weak bisimulation [22].

Lemma 2.12. *The labeled nets C and $\overline{E}$ are weakly bisimilar due to the agreement relation.*

Proof. First, we show that if we label each transition of C with itself in C and $\overline{E}$ and every transition $t \in (I_1 \setminus O_2) \uplus (I_2 \setminus O_1)$ in $\overline{E}$ with τ, then agreement between the markings of C and $\overline{E}$ is a weak bisimulation.

The initial markings m_C and $m_{\overline{E}}$ strongly agree by Definitions 2.4 and 2.5. Writing ϱ for the agreement relation, we now assume that $(m, \overline{m}) \in \varrho$. To prove that ϱ is a weak bisimulation, we have to show that

1. If $m \xrightarrow{t} m'$, then there exists $\overline{m}'$ such that $\overline{m} \stackrel{t}{\Longrightarrow} \overline{m}'$ and $(m', \overline{m}') \in \varrho$; and
2. If $\overline{m} \xrightarrow{t} \overline{m}'$, then there exists m' such that $m \stackrel{t}{\Longrightarrow} m'$ and $(m', \overline{m}') \in \varrho$.

Consider the first item. By firing all τ-labeled transitions of $\overline{E}$ that are enabled at $\overline{m}$, we can empty each place p^o while shifting the tokens to the respective place p^i. Let $\overline{m}''$ be the resulting marking of $\overline{E}$. Then $\overline{m} \stackrel{\varepsilon}{\Longrightarrow} \overline{m}''$ and $\overline{m}''$ strongly agrees with m, because firing a τ-labeled transition does not change the marking on the common places and no place p^o is marked now. By Proposition 2.10(2), we have $\overline{m}'' \xrightarrow{t} \overline{m}'$ such that m' and $\overline{m}'$ agree.

For the second item, we can set $m = m'$ if t is τ-labeled and, clearly, m' and $\overline{m}'$ agree then. Otherwise, we can conclude from Proposition 2.10(1) that marking m' exists such that m' and $\overline{m}'$ agree. Thus, $m \stackrel{t}{\Longrightarrow} m'$ and ϱ is a weak bisimulation.

Because the original labelings can be obtained by the same relabeling from the labelings considered in the first part of the proof, agreement is also a weak bisimulation for C and $\overline{E}$. □

We introduce a few, more compact notations. Given a set of traces, we define the prefix closure, suffix closure, and the remainder of this set. The suffix closure is also called the set of all *continuations* of a language.

Definition 2.13 (closures, remainder). Let $U \in \mathcal{P}(\Sigma^*)$. Then,

- $\downarrow U = \{u \in \Sigma^* \mid \exists v \in U : u \sqsubseteq v\}$ is the *prefix closure* of U.
- $\uparrow U = \{u \in \Sigma^* \mid \exists v \in U : v \sqsubseteq u\}$ is the *suffix closure* of U.
- $v^{-1}U = \{u \in \Sigma^* \mid vu \in U\}$ is the *remainder* of v in U.

For the suffix closure, we can show the following properties.

Lemma 2.14. *Let $X, Y \in \mathcal{P}(\Sigma^*)$. Then the following properties hold:*

1. $\uparrow (X \cup Y) = \uparrow X \cup \uparrow Y$
2. $x^{-1}(X \cup Y) = x^{-1}X \cup x^{-1}Y$
3. $y \notin \uparrow Y$ *implies* $y^{-1}(X \cup \uparrow Y) \subseteq \uparrow (y^{-1}(X \cup Y))$

Proof. Items (1) and (2) are trivial. For (3) observe that $y^{-1}(X \cup \uparrow Y) = y^{-1}X \cup y^{-1}\uparrow Y \subseteq \uparrow (y^{-1}X) \cup \uparrow (y^{-1}Y) = \uparrow (y^{-1}(X \cup Y))$. □

3 Unbounded nets and no final markings

In this section, we consider possibly unbounded open nets and ignore final markings. The resulting notions of responsiveness and r-accordance yield an equivalence, which is similar to P-deadlock equivalence in [33].

Definition 3.1 (responsiveness). Let N_1 and N_2 be composable open nets. A marking m of $N_1 \oplus N_2$ is *responsive* if we can reach from m a marking that enables a transition t with $t^\bullet \cap (O_1 \uplus O_2) \neq \emptyset$. The open nets N_1 and N_2 are *responsive* if their composition $N_1 \oplus N_2$ is a closed net and every reachable marking of $N_1 \oplus N_2$ is responsive.

Responsiveness ensures that at least one net can talk to the other repeatedly. This property depends on N_1 and N_2 in combination: In the composition, each of them will usually not reach all markings it could reach in other contexts; also, it suffices that just one component can enable an output transition. We are actually aiming at a setting with bounded open nets for which the term 'responsive' will imply mutual communication, see Sect. 6 for details.

Based on the correctness criterion responsiveness, we define an *r-controller* of an open net N as an open net C such that N and C are responsive.

Definition 3.2 (r-controller). An open net C is an *r-controller* of an open net N if N and C are responsive.

For every 'truly' (i.e., not closed) open net N, there exists an r-controller—the latter just has to continuously send a message, which is possible, because by Definition 2.3 N has at least one input place.

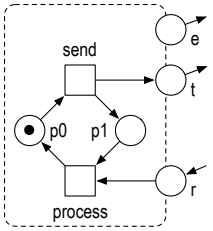


Fig. 3: The open net S' modeling a patched time server. We have $\Omega_{S'} = \{[]\}$.

If the r-controllers of an open net are a superset of the r-controllers of another open net, then the first open net is a refinement of the second; intuitively, it makes more users happy due to responsive interaction than the latter. We refer to the resulting refinement relation as *r-accordance*, which gives a necessary requirement for a refinement. For modular reasoning, a refinement relation should be a precongruence for composition. Because r-accordance shall turn out not to be one, we will make it stricter (smaller) as far as needed to obtain such a precongruence, and we already introduce a notation for this coarsest precongruence.

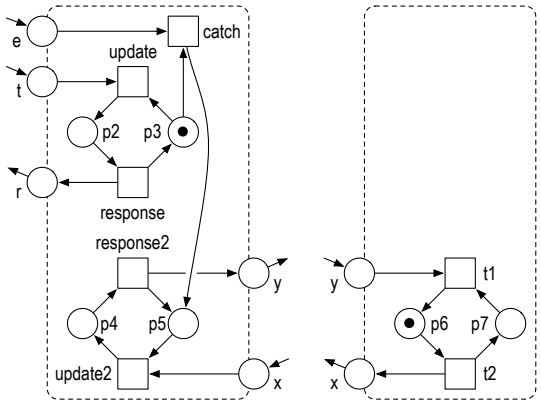
Definition 3.3 (r-accordance). For interface-equivalent open nets *Impl* and *Spec*, *Impl r-accords with Spec*, denoted by *Impl* $\sqsubseteq_{r,acc}$ *Spec*, if for all open nets C the following holds: If C is an r-controller of *Spec*, then C is also an r-controller of *Impl*.

We denote the *coarsest precongruence* w.r.t. $\oplus$ contained in $\sqsubseteq_{r,acc}$ by $\sqsubseteq^c_{r,acc}$.

Example 3.1. In Fig. 1, the open net S is an r-controller of the open net C, and vice versa: Either they can mutually communicate over the interface places t and r or C repeatedly produces a token on place r after consuming a token from e. The open net S' in Fig. 3 models a patched time server. It has the same functionality as the open net S, but it never sends an error message. The open net S' r-accords with the open net S: Every r-controller C of S must expect an error from S (i.e., a token on interface place e) and, thus, C is also an r-controller of S', where an error may never happen.

For an example that accordance does not guarantee compositionality, see Fig. 4. Although S' r-accords with S, $S' \oplus A$ does not r-accord with $S \oplus A$: The open net B is an r-controller of $S \oplus A$ but not of $S' \oplus A$, because the transition *catch* in $S' \oplus A$ can never fire and, thus, firing transition t_2 in B leads to a nonresponsive marking of $(S' \oplus A) \oplus B$.

In the following, we first give a trace-based characterization for r-accordance. Afterward, we characterize the coarsest precongruence that is contained in the r-accordance relation.



(a) Open net A (b) Open net B

Fig. 4: Two open nets proving that r-accordance (and fr-accordance, see Sect. 4) is not a precongruence with regard to open net composition $\oplus$. In addition to the models, we have $\Omega_A = \{[p_5]\}$ and $\Omega_B = \{[p_6]\}$.

3.1 A trace-based semantics for responsiveness

Our trace-based semantics of an open net N considers the set of *stop-traces* of its environment $env(N)$. A stop-trace records a run of $env(N)$ that ends in a marking weakly enabling actions of I only, such that N stops unless some input is provided. This trace is a weak version of a notion with the same name in [31,32], where only transitions of I and no τ-transitions are allowed to be enabled.

Definition 3.4 (stop-semantics). Let N be a labeled net. A marking m of N is a *stop except for inputs* if there is no $o \in \Sigma_{out}$ such that $m \stackrel{o}{\Longrightarrow}$. The *stop-semantics* of N is defined by the set of traces

$$stop(N) = \{w \mid m_N \stackrel{w}{\Longrightarrow} m \text{ and } m \text{ is a stop except for inputs}\}\,.$$

Example 3.2. Consider the open nets S and C in Fig. 1. The language of S is
$L(S) = \{w \in \{r,t\}^* \mid \forall u \sqsubseteq w : |u|_t \leq |u|_r + 1\}$
$\cup\, \{wev \mid w,v \in \{r,t\}^* \wedge \forall u \sqsubseteq w : |u|_t \leq |u|_r + 1 \wedge |wv|_t \leq |w|_r\}$
Observe that after firing e, transition r is continuously enabled in $env(S)$ while transition t may also fire because of pending tokens on the place t^o. Every stop-trace of S either contains an e or the number of r's is smaller than the number of t's; more precisely,
$stop(S) = \{w \in \{r,t\}^* \mid |w|_t = |w|_r + 1 \wedge \forall u \sqsubseteq w : |u|_t \leq |u|_r + 1\}$
$\cup\, \{wev \mid wev \in L(S)\}$
For C, every stop-trace has an equal number of t's and r's, and no stop-trace contains an e because of transition *reset*; more precisely,
$stop(C) = \{w \in \{r,t\}^* \mid |w|_t = |w|_r \wedge \forall v \sqsubseteq w : |v|_t \geq |v|_r\}$.

The presence of stop-traces in two open nets N_1 and N_2 is closely related to the question whether N_1 and N_2 are responsive. We first relate responsive markings and stops except for inputs.

Lemma 3.5 (responsive marking vs. stop except for inputs). *Let N_1 and N_2 be composable open nets such that $N_1 \oplus N_2$ is a closed net, and let $E = env(N_1) \| env(N_2)$. Let m be a marking of $N_1 \oplus N_2$ and $\overline{m}$ be a marking of E such that m and $\overline{m}$ agree. Then the following hold:*

1. *If m is responsive, then $\overline{m}$ is not a stop except for inputs.*
2. *If m and $\overline{m}$ strongly agree, the converse of (1) also holds.*

Proof. We have $N_1 \oplus N_2 = env(N_1 \oplus N_2)$ and $(I_1 \cap O_2) \uplus (I_2 \cap O_1) = O_1 \uplus O_2$ because $N_1 \oplus N_2$ is a closed net. Let $C = N_1 \oplus N_2$ and $O = O_1 \uplus O_2$.

$\Rightarrow$: W.l.o.g., assume that m and $\overline{m}$ strongly agree; otherwise, $\overline{m}$ is no stop except for inputs as there exists an $o \in O$ with $\overline{m} \xrightarrow{o}$ by Definition 2.9, hence $\overline{m} \stackrel{o}{\Longrightarrow}$. As m is responsive, we can fire some vt in C such that t is the first transition produces a token on some $x \in O$, i.e., $m \xrightarrow{vt} m'$ in C. Then it is possible to insert transitions from O of E into v such that for the resulting $v't$: $\overline{m} \xrightarrow{v't} \overline{m}'$ in E and also m' and $\overline{m}'$ strongly agree by Proposition 2.11(1). Hence either $\overline{m} \stackrel{y}{\Longrightarrow}$ for one of the inserted transitions y or $\overline{m} \stackrel{x}{\Longrightarrow}$, and $\overline{m}$ is not a stop except for inputs.

$\Leftarrow$: Because $\overline{m}$ is no stop except for inputs but does not enable any transition $x \in O$ (by strong agreement), we have $\overline{m} \xrightarrow{v} \overline{m}' \xrightarrow{t} \overline{m}''$ in E where neither t nor any transition in v is in O, and $\overline{m}''$ enables a transition $x \in O$ disabled at $\overline{m}'$. Hence, $x^o \in t^\bullet$ in E and, consequently, $x \in t^\bullet$ in C by Definition 2.5. Applying Proposition 2.11(2), we get $m \xrightarrow{v} m'$ in C such that m' and $\overline{m}'$ agree. Thus, transition t is enabled at m' in C, and m is responsive. □

Next, we relate a stop except for inputs in the parallel composition of two environments to a stop except for inputs in one of the involved environments.

Lemma 3.6 (stop except for inputs vs. stop-semantics). *Let N_1 and N_2 be composable open nets, and let $E = env(N_1) \| env(N_2)$. Let m_1 and m_2 be markings of $env(N_1)$ and $env(N_2)$, respectively. Then, $m = m_1 + m_2$ is a stop except for inputs in E iff m_1 and m_2 are stops except for inputs.*

Proof. $\Rightarrow$: W.l.o.g., assume that m_1 is not a stop except for inputs due to $m_1 \stackrel{o}{\Longrightarrow}$ with $o \in O_1$. As m_2 enables $o \in I_2$, we get $m \stackrel{o}{\Longrightarrow}$ with $o \in O_1 \uplus O_2$ by Proposition 2.8, hence m is no stop except for inputs.

$\Leftarrow$: Because m_1 and m_2 are stops except for inputs, there is no $o \in O_1 \uplus O_2$ such that $m_1 \stackrel{o}{\Longrightarrow}$ in $env(N_1)$ and $m_2 \stackrel{o}{\Longrightarrow}$ in $env(N_2)$. Applying Proposition 2.8, $m_1 + m_2 \stackrel{o}{\Longrightarrow}$ is not in E; thus, m is a stop except for inputs. □

The following proposition combines Lemma 3.5 and Lemma 3.6, thereby relating responsiveness and the *stop*-semantics.

Proposition 3.7 (responsiveness vs. *stop*-semantics). *Let N_1 and N_2 be composable open nets such that $N_1 \oplus N_2$ is a closed net. Then*

$$N_1 \text{ and } N_2 \text{ are responsive} \quad \text{iff} \quad \mathit{stop}(N_1) \cap \mathit{stop}(N_2) = \emptyset\,.$$

Proof. Let $C = N_1 \oplus N_2$ and $E = \mathit{env}(N_1) \| \mathit{env}(N_2)$.

$\Rightarrow$: Proof by contraposition. Assume a trace $w \in \mathit{stop}(N_1) \cap \mathit{stop}(N_2)$. Hence, $m_{\mathit{env}(N_1)} \stackrel{w}{\Longrightarrow} m_1$ in $\mathit{env}(N_1)$ and $m_{\mathit{env}(N_2)} \stackrel{w}{\Longrightarrow} m_2$ in $\mathit{env}(N_2)$ such that both m_1 and m_2 are stops except for inputs. Applying Proposition 2.8, we have $m_E \stackrel{w}{\Longrightarrow} m_1 + m_2$ in E and, by Lemma 3.6, $m_1 + m_2$ is a stop except for inputs. Markings m_E and m_C strongly agree by Definitions 2.4 and 2.5. By Proposition 2.11(2), a marking m is reachable from m_C in C such that $m_1 + m_2$ and m agree, and, by Lemma 3.5, m is not responsive. Thus, N_1 and N_2 are not responsive.

$\Leftarrow$: Proof by contraposition. Assume a marking m is reachable in C such that m is not responsive. Markings m_E and m_C strongly agree by Definitions 2.4 and 2.5. Applying Proposition 2.11(1), we have $m_E \stackrel{w}{\Longrightarrow} m_1 + m_2$ in E for some w such that m and $m_1 + m_2$ strongly agree, and, by Lemma 3.5, $m_1 + m_2$ is a stop except for inputs. By Proposition 2.8, we have $m_{\mathit{env}(N_1)} \stackrel{w}{\Longrightarrow} m_1$ in $\mathit{env}(N_1)$ and $m_{\mathit{env}(N_2)} \stackrel{w}{\Longrightarrow} m_2$ in $\mathit{env}(N_2)$, and, by Lemma 3.6, m_1 and m_2 are stops except for inputs. Thus, $w \in \mathit{stop}(N_1) \cap \mathit{stop}(N_2)$. □

Example 3.3. For the open nets S and C in Fig. 1, the stop-traces are given in Example 3.2. One can see that $\mathit{stop}(S) \cap \mathit{stop}(C) = \emptyset$; thus, S and C are indeed responsive, as already claimed in Example 3.1.

The next theorem provides a trace-based characterization of r-accordance as inclusion of stop-traces.

Theorem 3.8 (r-accordance and stop inclusion coincide). *For two interface-equivalent open nets Impl and Spec, we have*

$$\mathit{Impl} \sqsubseteq_{r,acc} \mathit{Spec} \quad \text{iff} \quad \mathit{stop}(\mathit{Impl}) \subseteq \mathit{stop}(\mathit{Spec})\,.$$

Proof. $\Leftarrow$: Proof by contraposition. Consider an open net C such that $\mathit{Impl} \oplus C$ and, by interface equivalence, $\mathit{Spec} \oplus C$ are closed nets. Assume that C is not an r-controller of *Impl*. Then *Impl* and C are not responsive by Definition 3.2, and there exists a trace $w \in \mathit{stop}(\mathit{Impl}) \cap \mathit{stop}(C)$ by Proposition 3.7. Because of stop-inclusion, we have $w \in \mathit{stop}(\mathit{Spec}) \cap \mathit{stop}(C)$. Again with Proposition 3.7, we see that *Spec* and C are not responsive; that is, C is not an r-controller of *Spec*.

$\Rightarrow$: The idea is to construct for a stop-trace w of *Impl* a net and show using the accordance of *Impl* and *Spec* that w is also a stop-trace of *Spec*.

Let I be the input and O be the output places of *Impl* and of *Spec*. Let $w \in \mathit{stop}(\mathit{Impl})$ and $w = w_1 \dots w_n$ with $w_j \in I \uplus O$, for $j = 1, \dots, n$. Define the open net $N_w = (P, T, F, m_0, O, I, \emptyset)$ by

- $P = \{p_0, \dots, p_n\}$,

- $T = \{t_1, \ldots, t_n\}$,
- $F = \quad \{(p_i, t_{i+1}) \mid 0 \leq i \leq n-1\}$
  $\uplus\ \{(t_i, p_i) \mid 1 \leq i \leq n\}$
  $\uplus\ \{(w_i, t_i) \mid 1 \leq i \leq n, w_i \in O\}$
  $\uplus\ \{(t_i, w_i) \mid 1 \leq i \leq n, w_i \in I\}$, *and*
- $m_0 = [p_0]$.

W.l.o.g., we assume $I \neq \emptyset$. Otherwise, $I = O = \emptyset$ by Definition 2.3 and, therefore, $\mathit{stop}(\mathit{Impl}) = \{\varepsilon\} \subseteq \{\varepsilon\} = \mathit{stop}(\mathit{Spec})$ by Definitions 2.5 and 3.4.

Let $o \in I$ be arbitrary but fixed. We extend N_w to an open net $N_{w,o} = (P', T', F', m'_0, O, I, \Omega)$—see Fig. 5—with

- $P' = P \uplus \{p, p'\} \uplus \{p'_0, \ldots, p'_{n-1}\}$,
- $T' = T \uplus \{t, t'_0, \ldots, t'_{n-1}, t''_0, \ldots, t''_{n-1}\} \uplus \{t_{w_i} \mid w_i \in O\}$,
- $F' = \quad F$
  $\uplus\ \{(p', t), (t, p'), (t, o)\}$
  $\uplus\ \{(p, t_{w_i}) \mid w_i \in O\}$
  $\uplus\ \{(t_{w_i}, p') \mid w_i \in O\}$
  $\uplus\ \{(w_i, t_{w_i}) \mid w_i \in O\}$
  $\uplus\ \{(p_i, t'_i) \mid 0 \leq i \leq n-1\}$
  $\uplus\ \{(t'_i, p'_i) \mid 0 \leq i \leq n-1\}$
  $\uplus\ \{(p'_i, t''_i) \mid 0 \leq i \leq n-1\}$
  $\uplus\ \{(t''_i, p'_i) \mid 0 \leq i \leq n-1\}$
  $\uplus\ \{(t''_i, o) \mid 0 \leq i \leq n-1\}$,
- $m'_0 = [p_0, p]$, and
- $\Omega = \{[p_n, p]\}$.

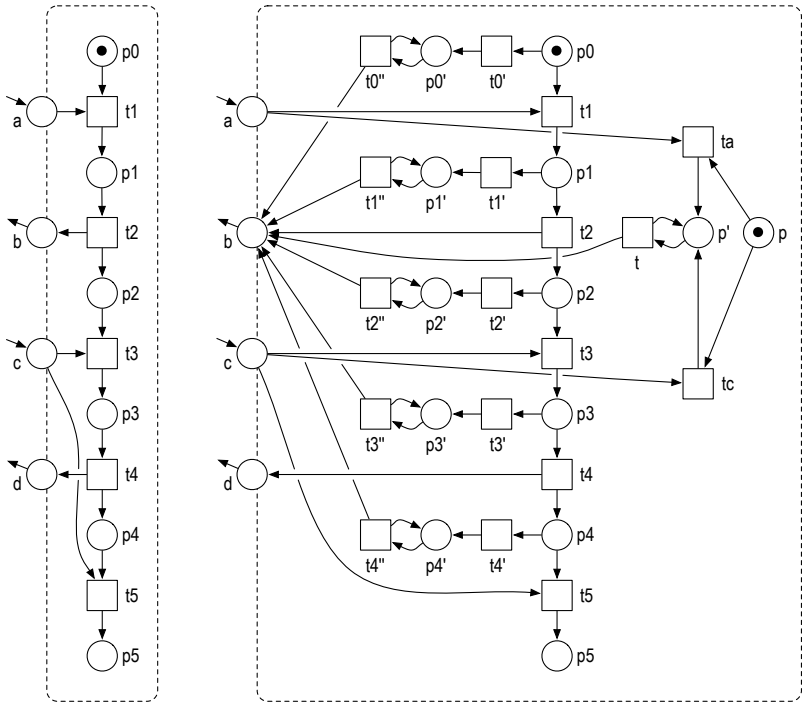
At a stop except for inputs of $\mathit{env}(N_{w,o})$, no transition t with $o \in t^\bullet$ (in $N_{w,o}$) is enabled or can be enabled by firing τ-labeled transitions of $\mathit{env}(N_{w,o})$ by Definition 3.4. Hence, a marking of $\mathit{env}(N_{w,o})$ is a reachable stop except for inputs if and only if it is the marking $[p_n, p]$ (1)—keep in mind that every $a \in I$ is an output place of $N_{w,o}$.

Obviously, $\mathit{Impl} \oplus N_{w,o}$ as well as $\mathit{Spec} \oplus N_{w,o}$ are closed nets by construction of $N_{w,o}$. Because $w \in \mathit{stop}(N_{w,o})$ according to observation (1), *Impl* and $N_{w,o}$ are not responsive by Proposition 3.7 and choice of w. Hence, $N_{w,o}$ is not a controller of *Impl* by Definition 3.2, and neither a controller of *Spec*, as *Impl* accords with *Spec*. We conclude that *Spec* and $N_{w,o}$ are not responsive because of Definition 3.2. Again with Proposition 3.7 and Definition 3.4, there exists a $v \in (I \uplus O)^*$ such that

$$m_{\mathit{env}(\mathit{Spec})} \stackrel{v}{\Longrightarrow} m_1 \text{ and } m_{\mathit{env}(N_{w,o})} \stackrel{v}{\Longrightarrow} m_2\,, \qquad (2)$$

where both m_1 and m_2 are stops except for inputs.

According to observation (1), transitions $t_1, \ldots, t_n$ of $N_{w,o}$ occur in this order in a run u of $\mathit{env}(N_{w,o})$ underlying v and, thus, there is no occurrence of a transition t'_j in u by construction. Furthermore, no transition t_{w_i} has fired and



(a) N_w (b) $N_{w,b}$

Fig. 5: Construction of N_w and $N_{w,b}$ for $w = abcdc$ with $b, d \in I$ and $a, c \in O$

removed the token from p. These facts imply that the Parikh vectors of w and v agree: Each t_i consumes a token from or produces a token on w_i, but all interface places are empty at the end of the traces.

In u, each occurrence of t_j with $w_j \in t_j^\bullet$ (as output place of $N_{w,o}$, i.e., $w_j \in I$) is paired with a succeeding occurrence of w_j (as transition of $env(N_{w,o})$); otherwise, transition w_j would be enabled at m_2 in $env(N_{w,o})$ and m_2 would not be a stop except for inputs. As transition w_j is not in conflict with any other transition of $env(N_{w,o})$, we assume that w_j fires immediately after t_j. In the corresponding rearranged trace v' of v, all $w_j \in I$ occur in the same order as in w, and v' still leads to m_2.

Similarly, each occurrence of t_j with $w_j \in {}^\bullet t_j$ (as input place of $N_{w,o}$, i.e., $w_j \in O$) is paired with a preceding occurrence of w_j (as transition of $env(N_{w,o})$), which can be delayed such that it occurs immediately before t_j. In the corresponding rearranged trace v'' of v', all $w_j \in O$ occur in the same order as in w, because v' and v'' have the same Parikh vector as w; thus, v'' is w.

We have transformed v to w by moving $w_j \in I$ backwards and $w_j \in O$ forwards. This can also be done in the run underlying v in $env(Spec)$, because

the respective transitions have an empty preset and postset, respectively. Thus,

$$m_{env(Spec)} \stackrel{w}{\Longrightarrow} m_1 \text{ (and } m_{env(N_{w,o})} \stackrel{w}{\Longrightarrow} m_2) \qquad (3)$$

and therefore $w \in stop(Spec)$. □

Example 3.4. For the open net S' in Fig. 3, we have
$stop(S') = \{w \in \{r,t\}^* \mid |w|_t = |w|_r + 1 \wedge \forall v \sqsubseteq w : |v|_t \leq |v|_r + 1\} \subseteq stop(S)$
(see Example 3.2). We conclude with Theorem 3.8 that S' r-accords with S.

Accordance, as defined in Definition 3.3, does not guarantee compositionality; that is, it is not a precongruence w.r.t. open net composition $\oplus$. We showed this in Example 3.1 using open nets S and S'. To see the difference between S and S' consider any trace of $env(S)$ not containing e. In the marking reached, there is a token on p_0, p_1, or e^o and the trace re is always possible. In contrast, $env(S')$ can always refuse re, because $env(S')$ can never perform an e. Therefore, it is not possible to differentiate between S and S' with something even weaker than standard failure semantics, as introduced by [7] (like the trace semantics we employed in Definition 3.4).

3.2 Deriving the coarsest precongruence for responsiveness

Taking into account the example in Fig. 4 and the observation that refusal information is necessary to distinguish open nets in terms of r-accordance, we shall characterize the precongruence $\sqsubseteq^c_{r,acc}$ in terms of a variant of failure semantics. For this, we will not use CSP failures, as introduced by Brookes et al. [7], but Vogler's $\mathcal{F}^+$-semantics [33]. Whereas a *failure* is a pair (w, X) where w is a trace of a net and X is a subset of the alphabet—a *refusal set*—the $\mathcal{F}^+$-semantics is a stronger notion, considering pairs (w, X) where X is a set of traces; such a pair is a *tree failure*.

Definition 3.9 ($\mathcal{F}^+$-semantics). The *$\mathcal{F}^+$-semantics* of a labeled net N is

$$\mathcal{F}^+(N) = \{(w, X) \in \Sigma^* \times \mathcal{P}(\Sigma^+) \mid \exists m \in M_N : m_N \stackrel{w}{\Longrightarrow} m \ \wedge \ \nexists w' \in X : m \stackrel{w'}{\Longrightarrow}\}.$$

We say that after executing w, N *refuses* X.

Example 3.5. We can distinguish the open nets S and S' in Fig. 1a and Fig. 3 by their $\mathcal{F}^+$-semantics: We have $(\varepsilon, r^*e) \in \mathcal{F}^+(S')$ (because no trace of S' contains an e) but $(\varepsilon, r^*e) \notin \mathcal{F}^+(S)$ (because we cannot prevent trace re after trace ε, as explained below Example 3.4).

If we consider the composition $N_1 \oplus N_2$ of two open nets N_1 and N_2, then its $\mathcal{F}^+$-semantics coincides with that of the parallel composition of the two environments, $env(N_1) \Uparrow env(N_2)$.

Lemma 3.10. *For two composable open nets N_1 and N_2, we have*

$$\mathcal{F}^+(env(N_1 \oplus N_2)) = \mathcal{F}^+(env(N_1) \Uparrow env(N_2))\,.$$

Proof. Follows directly from Lemma 2.12: If one net has a tree failure (w, X) due to a marking m, then the other net can reach an agreeing marking m' via the trace w. If some trace $v \in X$ could be performed from m', this would also be possible from m due to weak bisimilarity, yielding a contradiction. Thus, (w, X) is also a tree failure of the other net. □

Before determining the $\mathcal{F}^+$-semantics for the composition of two open nets, we first recall how this can be done for the composition of two labeled nets [33, Theorem 3.3.15] and for the hiding of common actions [33, Theorem 3.4.2]; we recall this to prepare the next section, where we study a new variation of the $\mathcal{F}^+$-semantics.

Proposition 3.11 ($\mathcal{F}^+$-semantics for labeled net composition [33]). *For two composable labeled nets N_1 and N_2, we have*

$$\begin{aligned}\mathcal{F}^+(N_1 \| N_2) = \{(w, X) \mid \exists (w_1, X_1) \in \mathcal{F}^+(N_1), (w_2, X_2) \in \mathcal{F}^+(N_2) :\\ w \in w_1 \| w_2 \wedge \forall x \in X :\\ x \in x_1 \| x_2 \textit{ implies } x_1 \in X_1 \vee x_2 \in X_2\}\,.\end{aligned}$$

In the next proposition recapitulates [33, Theorem 3.4.2]. We consider a labeled net N/A, $A \subseteq \Sigma$ and use $\phi(w)$ to denote $w|_{\Sigma \setminus A}$. We canonically extend the notion of $\phi(w)$ pointwise to sets of traces.

Proposition 3.12 ([33]). *For any labeled net N and any label set $A \subseteq \Sigma_N^*$, we have*

$$\mathcal{F}^+(N/A) = \{(\phi(w), X) \mid (w, \phi^{-1}(X)) \in \mathcal{F}^+(N)\}\,.$$

We now combine Lemma 3.10 and Propositions 3.12 and 3.11 to show how the $\mathcal{F}^+$-semantics for the composition of two open nets can be determined.

Proposition 3.13 ($\mathcal{F}^+$-semantics for open net composition). *For two composable open nets N_1 and N_2, we have*

$$\begin{aligned}\mathcal{F}^+(N_1 \oplus N_2) = \{(w, X) \mid \exists (w_1, X_1) \in \mathcal{F}^+(N_1), (w_2, X_2) \in \mathcal{F}^+(N_2) :\\ w \in w_1 \Uparrow w_2 \wedge \forall x \in X :\\ x \in x_1 \Uparrow x_2 \textit{ implies } x_1 \in X_1 \vee x_2 \in X_2\}\,.\end{aligned}$$

Proof. Proposition 3.11 shows that the right part of this equation—with $\|$ replacing $\Uparrow$—is equal to $\mathcal{F}^+(env(N_1) \| env(N_2))$; then, one can hide the common actions of $env(N_1)$ and $env(N_2)$, and by Proposition 3.12 the right hand side is equal to $\mathcal{F}^+(env(N_1) \Uparrow env(N_2))$; the latter is equal to $\mathcal{F}^+(env(N_1 \oplus N_2)) = \mathcal{F}^+(N_1 \oplus N_2)$ by Lemma 3.10. □

For the present setting, the tree failures used in the $\mathcal{F}^+$-semantics give too much information about the moment of choice in an open net. This information can be removed by closing up under an ordering over tree failures. The resulting modification of the $\mathcal{F}^+$-semantics yields the following refinement relation.

Definition 3.14 ($\mathcal{F}^+$-refinement). For two interface-equivalent labeled nets *Impl* and *Spec*, *Impl* $\mathcal{F}^+$*-refines Spec*, denoted by $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+} \mathit{Spec}$, if

$$\forall (w, X) \in \mathcal{F}^+(\mathit{Impl}) : \exists x \in \{\varepsilon\} \cup \downarrow X : (wx, x^{-1}X) \in \mathcal{F}^+(\mathit{Spec})\,.$$

For two interface-equivalent open nets *Impl* and *Spec*, we define $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+_{fin}} \mathit{Spec}$, if $\mathit{env}(\mathit{Impl}) \sqsubseteq_{\mathcal{F}^+_{fin}} \mathit{env}(\mathit{Spec})$.

Example 3.6. We have $(\varepsilon, r^*e) \in \mathcal{F}^+(S')$ (see Example 3.5) but for all $x \in \{\varepsilon\} \cup \downarrow r^*e$, $(x, x^{-1}r^*e) \notin \mathcal{F}^+(S)$ because $r^{-1}r^*e = r^*e$. Thus, S' does not $\mathcal{F}^+$-refine S.

Using $\mathcal{F}^+$-semantics and -refinement is a technically very beneficial reformulation [29] of $\mathcal{F}^{++}$-inclusion in [33]. Our definition of $\mathcal{F}^+$-refinement is equivalent to the definition of the refinement relation $\sqsubseteq_{\mathcal{F}^+}$ in [29], which coincides with should (or fair) testing [6,29] as proved in [29, Theorem 36]. Should testing is a precongruence for composition [29], and with the help of Lemma 3.10, we can show that it is also a precongruence for the composition operator $\oplus$.

Theorem 3.15 (precongruence). $\mathcal{F}^+$*-refinement is a precongruence for the composition operator* $\oplus$*.*

Proof. Let *Impl* and *Spec* be interface-equivalent open nets with $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+} \mathit{Spec}$, and let C be an open net composable with both. We have to show, $\mathit{Impl} \oplus C \sqsubseteq_{\mathcal{F}^+} \mathit{Spec} \oplus C$.

We have $\mathcal{F}^+(\mathit{env}(\mathit{Spec} \oplus C)) = \mathcal{F}^+(\mathit{env}(\mathit{Spec}) \Uparrow \mathit{env}(C))$ by Lemma 3.10. Let A denote the common actions of $\mathit{env}(\mathit{Spec})$ and $\mathit{env}(C)$. Then we can replace operator $\Uparrow$ with operator $\|$ and make the hiding explicit, which results in $\mathcal{F}^+(\mathit{env}(\mathit{Spec}) \Uparrow \mathit{env}(C)) = \mathcal{F}^+((\mathit{env}(\mathit{Spec}) \| \mathit{env}(C))/A)$. Likewise, we derive $\mathcal{F}^+(\mathit{env}(\mathit{Impl} \oplus C)) = \mathcal{F}^+((\mathit{env}(\mathit{Impl}) \| \mathit{env}(C))/A)$.

Impl $\mathcal{F}^+$-refines *Spec* and in [29, Lemma 46] $\mathcal{F}^+$-refinement for labeled transition systems has been proved to be a precongruence for the composition operator $\|$, and it is also preserved under hiding of common actions [29, Lemma 45]. Thus, we obtain that $(\mathit{env}(\mathit{Impl}) \| \mathit{env}(C))/A \sqsubseteq_{\mathcal{F}^+} (\mathit{env}(\mathit{Spec}) \| \mathit{env}(C))/A$. As this only depends on the $\mathcal{F}^+$-semantics of the two nets, the equations above show $\mathit{Impl} \oplus C \sqsubseteq_{\mathcal{F}^+} \mathit{Spec} \oplus C$. □

The proof of Theorem 3.15 gives the following result.

Proposition 3.16. $\mathcal{F}^+$*-refinement is a precongruence for* $\|$ *and hiding on labeled nets.*

With the next theorem, we prove that $\mathcal{F}^+$-refinement is the coarsest precongruence that is contained in the r-accordance relation.

Theorem 3.17 (precongruence and $\mathcal{F}^+$-refinement coincide). *For two interface-equivalent open nets Impl and Spec, we have*

$$\mathit{Impl} \sqsubseteq_{\mathcal{F}^+} \mathit{Spec} \quad \textit{iff} \quad \mathit{Impl} \sqsubseteq^c_{r,acc} \mathit{Spec}\,.$$

Proof. $\Rightarrow$: In the following, we assume a trace $w \in \mathit{stop}(\mathit{Impl})$ and prove $w \in \mathit{stop}(\mathit{Spec})$. Then, applying Theorem 3.8, we get $\mathit{Impl} \sqsubseteq_{r,acc} \mathit{Spec}$, which in turn shows the claim with Theorem 3.15 and the definition of $\sqsubseteq^c_{r,acc}$.

So let O be the set of output places of *Impl* and, equivalently, of *Spec*. We have $w \in \mathit{stop}(\mathit{Impl})$ if and only if $(w, O) \in \mathcal{F}^+(\mathit{Impl})$ by Definition 3.4 and 3.9. Then, by $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+} \mathit{Spec}$, there must be a suitable $x \in \{\varepsilon\} \cup \downarrow O = \{\varepsilon\} \cup O$ that makes the defining condition of Definition 3.14 true. We cannot have $x \in O$ because $(wx, \{\varepsilon\}) \notin \mathcal{F}^+(\mathit{Spec})$ by Definition 3.9. Thus, $x = \varepsilon$ and $(w, O) \in \mathcal{F}^+(\mathit{Spec})$, implying $w \in \mathit{stop}(\mathit{Spec})$.

$\Leftarrow$: Let $(w, X) \in \mathcal{F}^+(\mathit{Impl})$. In addition, consider an open net C with the new output x and the new input y. Open net C has the empty initial marking, no final marking, and contains only a single transition that can indefinitely repeat to produce a token in x while consuming a token from y. The idea is to construct an open net N from (w, X) such that C is not an r-controller of $\mathit{Impl} \oplus N$ because of (w, X). For the moment, assume such a net N has been constructed. By $\mathit{Impl} \sqsubseteq^c_{r,acc} \mathit{Spec}$ and because $\sqsubseteq^c_{r,acc}$ is a precongruence, we have $\mathit{Impl} \oplus N \sqsubseteq^c_{r,acc} \mathit{Spec} \oplus N$ and thus $\mathit{Impl} \oplus N \sqsubseteq_{r,acc} \mathit{Spec} \oplus N$ by Definition 3.3. Thus, C is also not an r-controller of $\mathit{Spec} \oplus N$, and from this we will conclude that (w, X) is covered by $\mathcal{F}^+(\mathit{Spec})$ according to Definition 3.14. Then we will have proved $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+} \mathit{Spec}$.

As to the actual construction of N: The open net N has inputs $I = O_{\mathit{Impl}} \uplus \{x\}$ and outputs $O = I_{\mathit{Impl}} \uplus \{y\}$ and enables a transition sequence $v = t_1 \dots t_k$. Each transition in v is connected to an interface place of N such that the corresponding trace of interface actions is w; that is, N contains net N_w as in Fig. 5 (except that p_k is called p_ε here). Thus, we can essentially fire the trace w of $\mathit{env}(N)$ in $\mathit{Impl} \oplus N$ and, therefore, in $\mathit{Impl} \oplus N \oplus C$ by firing v instead of the labeled transitions. This way, we reach in *Impl* the marking m that refuses X in $\mathit{env}(\mathit{Impl})$; in N, there is only one token in the place p_ε and the token in the place p has been consumed. This token is necessary to enable a transition t' that is essential for responsiveness; that is, transitions t and t' would repeatedly communicate with C. The place p can only be marked again by firing some transition t'_z with $z \in X$, and this in turn requires the firing of a transition sequence that—similarly to v—looks to *Impl* (or rather $\mathit{env}(\mathit{Impl})$) like trace z. But this trace cannot be fired at m; hence, C is not an r-controller of $\mathit{Impl} \oplus N$.

To achieve the effect just described, the second part of the open net N encodes the tree part X of tree failure (w, X). Common prefixes thereby correspond to the same path in the X-part of N. When reaching an element of X, a token can be produced in the place p. Figure 6 illustrates this construction; it is a small adaptation of a construction that is used in [33, Fig. 3.19].

Let $w = w_1 \dots w_k$ such that for $j = 1, \dots, k$, $w_j \in I_{\mathit{Impl}} \uplus O_{\mathit{Impl}}$. Define the open net $N = (P, T, F, m_N, O, I, \emptyset)$ by

- $P = \{p\} \uplus \{p_i \mid 0 \le i \le k-1\} \uplus \{p_u \mid u \in \downarrow X \cup \{\varepsilon\}\}$

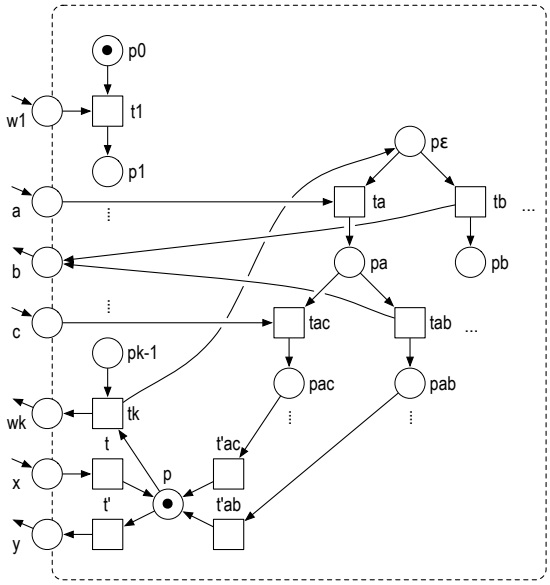


Fig. 6: Illustration of the construction of open net N.

- $T = \{t, t'\}$
  $\uplus\ \{t_i \mid 1 \leq i \leq k\}$
  $\uplus\ \{t_u \mid u \in {\downarrow} X \wedge u \neq \varepsilon\}$
  $\uplus\ \{t'_z \mid z \in X\}$
- $F = \{(p_i, t_{i+1}) \mid 0 \leq i \leq k-1\}$
  $\uplus\ \{(t_i, p_i) \mid 1 \leq i \leq k-1\}$
  $\uplus\ \{(x,t), (t,p), (p,t'), (t',y), (p,t_k), (t_k, p_\varepsilon)\}$
  $\uplus\ \{(p_u, t_{ua}) \mid a \in I_{Impl} \uplus O_{Impl} \wedge ua \in {\downarrow} X\}$
  $\uplus\ \{(t_u, p_u) \mid u \in {\downarrow} X \wedge u \neq \varepsilon\}$
  $\uplus\ \{(p_z, t'_z), (t'_z, p) \mid z \in X\}$
  $\uplus\ \{(w_i, t_i) \mid 1 \leq i \leq k \wedge w_i \in O_{Impl}\}$
  $\uplus\ \{(t_i, w_i) \mid 1 \leq i \leq k \wedge w_i \in I_{Impl}\}$
  $\uplus\ \{(a, t_{ua}) \mid a \in O_{Impl} \wedge ua \in {\downarrow} X\}$
  $\uplus\ \{(t_{ua}, a) \mid a \in I_{Impl} \wedge ua \in {\downarrow} X\}$, and
- $m_N = [p_0, p]$.

As argued previously, we now have that C is not an r-controller of $\mathit{Spec} \oplus N$; that is, a marking m_1 can be reached in $\mathit{Spec} \oplus N \oplus C$ where responsiveness is violated. Clearly, p must be empty in m_1; thus, v has been fired in N and possibly also some transitions in its X-part. There is only one token in the places of $\mathit{inner}(N)$, and it is in some p_u with $u \in {\downarrow} X \cup \{\varepsilon\}$. Let m_2 be the projection of m_1 onto the places of *Spec*. From the point of view of *Spec*, we have fired a trace wu of $\mathit{env}(\mathit{Spec})$, reaching m_2. Because in $\mathit{Spec} \oplus N \oplus C$ no $t'_{uu'}$ can become enabled (otherwise, C would be an r-controller), u' cannot be

fired in $env(Spec)$ at m_2. Thus, $(wu, \{u' \mid uu' \in X\}) \in \mathcal{F}^+(Spec)$ and, therefore, $Impl \sqsubseteq_{\mathcal{F}^+} Spec$. □

Example 3.7. We already showed with Fig. 4 that for the open nets S and S' $S' \sqsubseteq^c_{r,acc} S$ does not hold. This is now confirmed with Theorem 3.17 because S' does not $\mathcal{F}^+$-refine S by Example 3.6.

4 Unbounded nets and final markings

In this section, we consider possibly unbounded open nets and final markings. We refer to the resulting variant of responsiveness as *final-responsiveness* or *f-responsiveness* for short.

Definition 4.1 (*f*-responsiveness). Let N_1 and N_2 be composable open nets. A marking m of $N_1 \oplus N_2$ is *f-responsive* if either m is responsive or we can reach a final marking of $N_1 \oplus N_2$ from m. Open nets N_1 and N_2 are *f-responsive* if their composition $N_1 \oplus N_2$ is a closed net and every reachable marking in $N_1 \oplus N_2$ is f-responsive.

The notion of f-responsiveness generalizes responsiveness defined in Definition 3.1. While responsiveness requires at least one net of the composition to repeatedly talk to the other net, f-responsiveness also allows the composition to terminate instead—that is, to reach a common final marking.

Next, we redefine the notion of a controller and accordance for this variant of responsiveness which yields *fr*-controllers and *fr*-accordance. Again, because *fr*-accordance will turn out not to be a precongruence, we also introduce its coarsest precongruence.

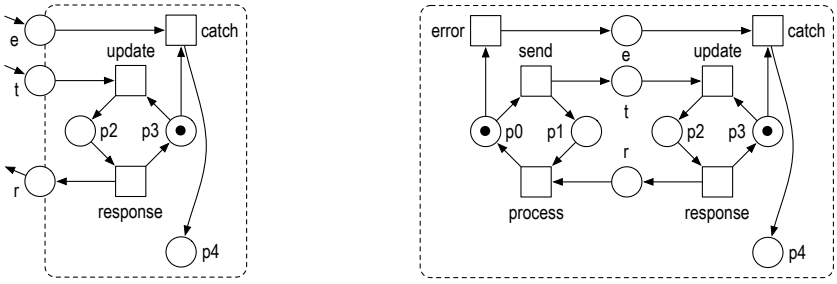
As for responsiveness, for every open net N, there exists an *fr*-controller—again, this open net just has to continuously send a message.

Definition 4.2 (*fr*-controller, *fr*-accordance). An open net C is an *fr-controller* of an open net N if N and C are f-responsive.

For two interface-equivalent open nets *Impl* and *Spec*, *Impl fr-accords with Spec*, denoted by $Impl \sqsubseteq_{fr,acc} Spec$, if for all open nets C: C is an *fr*-controller of *Spec* implies C is an *fr*-controller of *Impl*.

We denote the *coarsest precongruence* w.r.t. $\oplus$ contained in $\sqsubseteq_{fr,acc}$ by $\sqsubseteq^c_{fr,acc}$.

Example 4.1. The open net C' in Fig. 7a represents another client for the unreliable time server S in Fig. 1a. The client C' repeatedly updates its system time and responds with a response packet. However, if the time server sends an error message, C' receives this message (input place e) and terminates (final marking $[p_4]$). The open nets S and C' are composable; their composition $S \oplus C'$ is a closed net, which is depicted in Fig. 7b. C' is not an r-controller of S, because the nonresponsive marking $[p_4]$ is reachable in their composition $S \oplus C'$. In contrast, C' is an *fr*-controller of S because $[p_4]$ is a final marking of $S \oplus C'$.



(a) Open net C' (b) Open net $S \oplus C'$

Fig. 7: Open net C' modeling a terminating client for the open net S in Fig. 1a and their composition $S \oplus C'$. In addition to the models, we have $\Omega_{C'} = \Omega_{S \oplus C'} = \{[p_4]\}$.

The open net S' in Fig. 3 *fr*-accords with the open net S in Fig. 1a. Every *fr*-controller C of S must provide a token on r for each token on t; otherwise, S can get stuck in a nonfinal marking with a token on p_1. (Additionally, the *fr*-controller must be able to consume a token from e and reach a final marking.) Thus, S' and C are responsive and C is an *fr*-controller of S'.

Like r-accordance, also *fr*-accordance does not guarantee compositionality; that is, it is not a precongruence with respect to open net composition $\oplus$. Consider again Fig. 4 for a counterexample. The open net S' *fr*-accords with the open net S, but $S' \oplus A$ does not *fr*-accord with $S \oplus A$, because the open net B is an *fr*-controller of $S \oplus A$ but not of $S' \oplus A$: The only final marking $[p_5, p_6]$ is not reachable in $(S' \oplus A) \oplus B$, and the trace x is a stop-trace of both $S' \oplus A$ and B.

We continue by first giving a trace-based characterization for *fr*-accordance. Afterward, we characterize the coarsest precongruence that is contained in the *fr*-accordance relation.

4.1 A trace-based semantics for final-responsiveness

We extend Definition 3.4 by a set of *dead-traces*, a weak version of a notion with the same name in [31,32]. A dead-trace is a stop-trace leading to a nonfinal stop except for inputs—that is, a marking that is *dead except for inputs*.

Definition 4.3 (*stop dead*-semantics). Let N be a labeled net. A marking m of N is *dead except for inputs* if m is a stop except for inputs and there exists no final marking m' of N with $m \stackrel{\varepsilon}{\Longrightarrow} m'$. The *stop dead-semantics* of N is defined by the sets of traces

- $stop(N)$ and
- $dead(N) = \{w \mid m_N \stackrel{w}{\Longrightarrow} m \text{ and } m \text{ is dead except for inputs}\}$.

Example 4.2. For the open nets S and C' in Fig. 1a and 7a, $[\,]$ is the only final marking of S and $[p_4]$ is the only final marking C'. Therefore, we have

$$\begin{aligned} dead(S) = \quad & \{w \in \{r,t\}^* \mid |w|_t = |w|_r + 1 \wedge \forall v \sqsubseteq w : |v|_t \leq |v|_r + 1\} \\ & \cup \{wev \in L(S) \mid |wv|_t < |wv|_r\} \end{aligned}$$

Furthermore, we have

$$stop(C') = stop(C) \cup \{wev \mid w \in \{r,t\}^* \wedge v \in \{r,t,e\}^* \wedge \forall u \sqsubseteq wv : |u|_r \leq |u|_t\}$$

and

$$\begin{aligned} dead(C') = stop(C) \cup \{wev \mid\ & w \in \{r,t\}^* \wedge v \in \{r,t,e\}^* \wedge \forall u \sqsubseteq wv : \\ & |u|_r \leq |u|_t \wedge (|wv|_t > |wv|_r \vee |v|_e \geq 1)\} \end{aligned}$$

where $stop(C)$ has been defined in Example 3.2).

The presence of dead-traces in open nets N_1 and N_2 is closely related to the question whether N_1 and N_2 are f-responsive. We continue by relating first f-responsive and then the *stop dead*-semantics to markings that are dead except for inputs.

Lemma 4.4 (f-responsive vs. dead except for inputs). *Let N_1 and N_2 be composable open nets such that $N_1 \oplus N_2$ is a closed net. Let $E = env(N_1) \| env(N_2)$, and let m be a marking of $N_1 \oplus N_2$ and $\overline{m}$ be a marking of E such that m and $\overline{m}$ agree. If m is f-responsive, then $\overline{m}$ is not dead except for inputs. If m and $\overline{m}$ strongly agree, the converse also holds.*

Proof. We have $N_1 \oplus N_2 = env(N_1 \oplus N_2) =: C$ and $(I_1 \cap O_2) \uplus (I_2 \cap O_1) = O_1 \uplus O_2 =: O$, because $N_1 \oplus N_2$ is a closed net. Note that only transitions in O are not τ-labeled in E.

$\Rightarrow$: If m is responsive or if m and $\overline{m}$ do not strongly agree, we are done by Lemma 3.5. Otherwise, there is a final marking m' of C reachable from m. According to Proposition 2.11(1), there is a marking $\overline{m}'$ reachable from $\overline{m}$ in E such that m' and $\overline{m}'$ agree (even strongly). Marking $\overline{m}'$ is a final marking of E by Proposition 2.11(3). Thus, $\overline{m}$ is not dead except for inputs by Definition 4.3.

$\Leftarrow$: If $\overline{m}$ is not a stop except for inputs, then m is responsive by Lemma 3.5 and therefore f-responsive. Otherwise, there is a final marking $\overline{m}'$ of E reachable from $\overline{m}$. Applying Proposition 2.11(2), there is a marking m' reachable from m in C such that m' and $\overline{m}'$ agree. Marking m' is a final marking of C by Proposition 2.11(3), proving f-responsiveness of m. □

Lemma 4.5 (dead except for inputs vs. *stop dead*-semantics). *Let N_1 and N_2 be composable open nets, and let $E = env(N_1) \| env(N_2)$. Let m_1 and m_2 be markings of $env(N_1)$ and $env(N_2)$, respectively. Then, $m = m_1 + m_2$ is dead except for inputs in E iff m_1 is a stop except for inputs and m_2 is dead except for inputs, or vice versa.*

Proof. $\Rightarrow$: Because m is a stop except for inputs by Definition 4.3, both m_1 and m_2 are stops except for inputs by Lemma 3.6. Assume neither m_1 nor m_2 are dead except for inputs due to m_1' and m_2' respectively. Then $m = m_1 + m_2 \stackrel{\varepsilon}{\Longrightarrow} m_1' + m_2'$ by Proposition 2.8 and $m_1' + m_2'$ is a final marking by Proposition 2.11(3). This contradicts the assumption.

$\Leftarrow$: Due to Lemma 3.6, m is a stop except for inputs. W.l.o.g., assume m_2 is dead except for inputs. Whenever $m \stackrel{\varepsilon}{\Longrightarrow} m'$, Proposition 2.8 gives us $m_1 \stackrel{\varepsilon}{\Longrightarrow} m_1'$ and $m_2 \stackrel{\varepsilon}{\Longrightarrow} m_2'$ where neither m_2' nor—by Proposition 2.11(3)—$m' = m_1' + m_2'$ are final. Thus, m is dead except for inputs in E by Definition 4.3. □

We combine Lemma 4.4 and Lemma 4.5 and show how the *stop dead*-semantics can be used to characterize f-responsiveness.

Proposition 4.6 (f-responsiveness vs. *stop dead*-semantics). *Let N_1 and N_2 be composable open nets such that $N_1 \oplus N_2$ is a closed net. Then*

$$N_1 \text{ and } N_2 \text{ are } f\text{-responsive} \quad \text{iff} \quad \begin{array}{l} \mathit{stop}(N_1) \cap \mathit{dead}(N_2) = \emptyset \text{ and} \\ \mathit{dead}(N_1) \cap \mathit{stop}(N_2) = \emptyset\,. \end{array}$$

Proof. Let $C = N_1 \oplus N_2$ and $E = \mathit{env}(N_1) \| \mathit{env}(N_2)$.

$\Rightarrow$: Proof by contraposition. W.l.o.g., we assume a trace $w \in \mathit{stop}(N_1) \cap \mathit{dead}(N_2)$. Hence, $m_{\mathit{env}(N_1)} \stackrel{w}{\Longrightarrow} m_1$ in $\mathit{env}(N_1)$ and $m_{\mathit{env}(N_2)} \stackrel{w}{\Longrightarrow} m_2$ in $\mathit{env}(N_2)$ such that m_1 is a stop except for inputs and m_2 is dead except for inputs. By Lemma 4.5, $m_1 + m_2$ is dead except for inputs in E. By Proposition 2.11(2), a marking m is reachable in C such that m and $m_1 + m_2$ agree, and m is not f-responsive by Lemma 4.4.

$\Leftarrow$: Proof by contraposition. Assume $m_C \stackrel{\varepsilon}{\Longrightarrow} m$ in C such that m is not f-responsive. Applying Proposition 2.11(1), we can reach some $m_1 + m_2$ in E (with m_i a marking of N_i, $i = 1, 2$) such that m and $m_1 + m_2$ strongly agree, and, by Lemma 4.4, $m_1 + m_2$ is dead except for inputs. By Proposition 2.8, we have $m_{\mathit{env}(N_1)} \stackrel{w}{\Longrightarrow} m_1$ in $\mathit{env}(N_1)$ and $m_{\mathit{env}(N_2)} \stackrel{w}{\Longrightarrow} m_2$ in $\mathit{env}(N_2)$, and by Lemma 4.5, m_1 is a stop except for inputs and m_2 is dead except for inputs, or vice versa. Thus, $w \in \mathit{stop}(N_1) \cap \mathit{dead}(N_2)$ or $w \in \mathit{dead}(N_1) \cap \mathit{stop}(N_2)$. □

Example 4.3. Consider again Examples 3.2 and 4.2. One can see that $\mathit{stop}(S) \cap \mathit{dead}(C') = \emptyset$ and $\mathit{dead}(S) \cap \mathit{stop}(C') = \emptyset$; thus, C' is indeed an *fr*-controller of S, as already claimed in Example 4.1.

Inclusion of the stop- and dead-traces of open nets defines a refinement relation. With the next theorem, we prove that an open net *Impl* *fr*-accords with an open net *Spec* if and only if every stop-trace of *Impl* is contained in the stop-traces of *Spec* and every dead-trace of *Impl* is contained in the dead-traces of *Spec*. In other words, we provide a trace-based characterization of *fr*-accordance.

Theorem 4.7 (*fr*-accordance and *stop dead*-inclusion coincide). *For two interface-equivalent open nets Impl and Spec, we have*

$$\mathit{Impl} \sqsubseteq_{fr,acc} \mathit{Spec} \quad \text{iff} \quad \begin{array}{l} \mathit{stop}(\mathit{Impl}) \subseteq \mathit{stop}(\mathit{Spec}) \text{ and} \\ \mathit{dead}(\mathit{Impl}) \subseteq \mathit{dead}(\mathit{Spec})\,. \end{array}$$

Proof. $\Leftarrow$: Proof by contraposition. Consider an open net C such that $\mathit{Impl} \oplus C$ and, equivalently, $\mathit{Spec} \oplus C$ are closed nets. Assume that C is not an *fr*-controller of *Impl*. Then *Impl* and C are not f-responsive by Definition 4.2,

and we find a trace $w \in \mathit{stop}(\mathit{Impl}) \cap \mathit{dead}(C)$ or $w \in \mathit{dead}(\mathit{Impl}) \cap \mathit{stop}(C)$ by Proposition 4.6. Due to $\mathit{stop}\,\mathit{dead}$-inclusion, we have $w \in \mathit{stop}(\mathit{Spec}) \cap \mathit{dead}(C)$ and $w \in \mathit{dead}(\mathit{Spec}) \cap \mathit{stop}(C)$, respectively. Again with Proposition 4.6, *Spec* and C are not f-responsive; that is, C is not an *fr*-controller of *Spec*.

$\Rightarrow$: Let I be the input and O be the output places of *Impl* and, equivalently, of *Spec*. If $I = O = \emptyset$, we have $\mathit{stop}(\mathit{Impl}) = \{\varepsilon\} = \mathit{stop}(\mathit{Spec})$. Furthermore, either $\mathit{dead}(\mathit{Impl}) = \emptyset$ (and we are done) or $\mathit{dead}(\mathit{Impl}) = \{\varepsilon\}$ and we consider an open net C just consisting of a marked place, giving a final marking; C is not a controller of *Impl*, hence not of *Spec*, implying $\mathit{dead}(\mathit{Spec}) = \{\varepsilon\}$.

For the case $I \neq \emptyset \neq O$, we consider a trace $w \in \mathit{dead}(\mathit{Impl})$. Let $w = w_1 \dots w_n$ with $w_j \in I \uplus O$, for $j = 1, \dots, n$, and let $o \in I$ be arbitrary but fixed. We define an open net $N_{w,o} = (P', T', F', m_0', O, I, \Omega)$ exactly as in the proof of Theorem 3.8; see Fig. 5. As there, we see that a marking of $\mathit{env}(N_{w,o})$ is a reachable stop except for inputs if and only if it is the only final marking $[p_n, p]$ $(1')$, i.e., $\mathit{dead}(N_{w,o}) = \emptyset$.

Obviously, $\mathit{Impl} \oplus N_{w,o}$ as well as $\mathit{Spec} \oplus N_{w,o}$ are closed nets by construction of $N_{w,o}$. Because $w \in \mathit{stop}(N_{w,o})$ according to observation $(1')$, *Impl* and $N_{w,o}$ are not f-responsive by Proposition 4.6 and choice of w. Hence, $N_{w,o}$ is not an *fr*-controller of *Impl* and neither an *fr*-controller of *Spec*, because *Impl* *fr*-accords with *Spec*. Thus, *Spec* and $N_{w,o}$ are not f-responsive because of Definition 4.2. Again with Proposition 4.6 and Definition 4.3, there exists $v \in (I \uplus O)^*$ with

$$m_{\mathit{env}(\mathit{Spec})} \stackrel{v}{\Longrightarrow} m_1 \text{ and } m_{\mathit{env}(N_{w,o})} \stackrel{v}{\Longrightarrow} m_2 \qquad (2')$$

such that both m_1 and m_2 are stops except for inputs, and additionally m_1 or m_2 is dead except for inputs. As $\mathit{dead}(N_{w,o}) = \emptyset$, m_1 is dead except for inputs of $\mathit{env}(\mathit{Spec})$; furthermore, $m_2 = [p_n, p]$.

As in the proof of Theorem 3.8, we derive

$$m_{\mathit{env}(\mathit{Spec})} \stackrel{w}{\Longrightarrow} m_1 \text{ and } m_{\mathit{env}(N_{w,o})} \stackrel{w}{\Longrightarrow} m_2 \qquad (3')$$

and therefore $w \in \mathit{dead}(\mathit{Spec})$.

For a trace $w \in \mathit{stop}(\mathit{Impl})$, we fix some arbitrary $o \in I$ and define an open net $N'_{w,o} = (P', T', F', m_0', O, I, \emptyset)$, which is identical to $N_{w,o}$ except for its empty set of final markings. Thus, $w \in \mathit{dead}(N'_{w,o})$, and we succeed with an argumentation similar to the previous one. □

Example 4.4. In Example 3.4, we showed that $\mathit{stop}(S') \subseteq \mathit{stop}(S)$. Each $w \in \mathit{stop}(S')$ can reach the nonfinal marking $[p_1]$ in S' and in S, hence $\mathit{stop}(S') = \mathit{dead}(S')$ and $\mathit{stop}(S') \subseteq \mathit{dead}(S)$ (see Examples 3.4 and 4.2). Therefore, S' *fr*-accords with S.

As shown in Example 4.1, *fr*-accordance is not a precongruence with respect to open net composition $\oplus$. Therefore, we characterize the coarsest precongruence, which is contained in *fr*-accordance in the following.

4.2 Deriving the coarsest precongruence for final-responsiveness

The notion of f-responsiveness distinguishes between final and nonfinal markings. This information is needed to determine whether a marking is dead except for inputs. As we cannot derive this information from the $\mathcal{F}^+$-semantics, we must enhance it. The idea is basically to add an additional ingredient to a tree failure (w, X) yielding a triple (w, X, Y). This ingredient is *a set* Y, collecting traces that cannot lead the net to a final marking—including traces that cannot be performed at all. We bind the traces in Y to a certain marking m that is reached by executing w. Different markings m can be reached by w because of nondeterminism, so different sets Y may be assigned to them. This construction ensures that we can identify traces in $dead(N)$.

Definition 4.8 ($\mathcal{F}^+_{fin}$-semantics). The $\mathcal{F}^+_{fin}$*-semantics* of a labeled net N is a set of *fintree failures* and defined as

$$\begin{aligned}\mathcal{F}^+_{fin}(N) = \{(w, X, Y) \in \Sigma^* \times \mathcal{P}(\Sigma^+) \times \mathcal{P}(\Sigma^*) \mid \\ \exists m \in M_N : m_N \overset{w}{\Longrightarrow} m \\ \wedge\, \forall x \in X : m \overset{x}{\not\Longrightarrow} \\ \wedge\, \forall y \in Y : \forall m' : m \overset{y}{\Longrightarrow} m' \text{ implies } m' \notin \Omega_N\}\,.\end{aligned}$$

We say that after executing w, N *refuses* X and *fin-refuses* Y.

Example 4.5. We can distinguish the open nets S and S' in Fig. 1a and 3 by their $\mathcal{F}^+_{fin}$-semantics: We have $(\varepsilon, \emptyset, (tr)^*e) \notin \mathcal{F}^+_{fin}(S)$; that is, after executing ε reaching $[p_0]$, $[e]$, or $[p_1, t]$ we can always perform some $w \in (tr)^*r$ and reach a final marking. However, $(\varepsilon, \emptyset, (tr)^*e) \in \mathcal{F}^+_{fin}(S')$ because $env(S')$ can never perform e.

The $\mathcal{F}^+_{fin}$-refinement relation is similarly defined as the $\mathcal{F}^+$-refinement relation in Definition 3.14 by closing up under an ordering over the fintree failures in $\mathcal{F}^+_{fin}$; this removes the too detailed information about the moment of choice in an open net.

Definition 4.9 ($\mathcal{F}^+_{fin}$-refinement). For two interface-equivalent labeled nets *Impl* and *Spec*, *Impl* $\mathcal{F}^+_{fin}$*-refines Spec*, denoted by $Impl \sqsubseteq_{\mathcal{F}^+_{fin}} Spec$, if

$$\begin{aligned}&\forall (w, X, Y) \in \mathcal{F}^+_{fin}(Impl) : \\ &\qquad \exists x \in \{\varepsilon\} \cup {\downarrow} X \cup {\downarrow} Y : (wx, x^{-1}X, x^{-1}Y) \in \mathcal{F}^+_{fin}(Spec)\,.\end{aligned}$$

For two interface-equivalent open nets *Impl* and *Spec*, we define $Impl \sqsubseteq_{\mathcal{F}^+_{fin}} Spec$, if $env(Impl) \sqsubseteq_{\mathcal{F}^+_{fin}} env(Spec)$.

Example 4.6. We have $(\varepsilon, \emptyset, (tr)^*e) \in \mathcal{F}^+_{fin}(S')$ by Example 4.5. For all $w \in (tr)^*$, we have $(w, \emptyset, (tr)^*e) \notin \mathcal{F}^+_{fin}(S)$: After w, we reach any of the markings $[p_0]$, $[p_1, t]$, or $[e]$ from which we can always reach the final marking $[\,]$ with trace e or tre. Furthermore, for all $v \in (tr)^*t$, we have $(v, \emptyset, r(tr)^*e) \notin \mathcal{F}^+_{fin}(S)$: After v, we are in the marking $[p_1]$ from which we reach the final marking $[\,]$ with trace re. Thus, S' does not $\mathcal{F}^+_{fin}$-refine S.

We will now show that $\mathcal{F}^+_{fin}$-refinement is a precongruence for the composition operator $\oplus$. The proof will turn out to be fairly difficult. As in Sect. 3.2, we relate $\oplus$ to $\Uparrow$ on labeled nets and use that $\Uparrow$ is $\|$ followed by hiding of common actions.

If we consider the composition of two open nets N_1 and N_2, then its $\mathcal{F}^+_{fin}$-semantics coincides with that of the parallel composition of the two environments, $env(N_1)$ and $env(N_2)$.

Lemma 4.10. *For two composable open nets N_1 and N_2, we have*

$$\mathcal{F}^+_{fin}(env(N_1 \oplus N_2)) = \mathcal{F}^+_{fin}(env(N_1) \Uparrow env(N_2))\,.$$

Proof. This lemma follows directly from Lemma 2.12: If one net has a fintree failure (w, X, Y) due to a marking m, then the other net can reach an agreeing marking m' with the trace w. If a trace $x \in X$ could be performed from m' in the second net, this would also be possible from m in the first net due to bisimilarity, yielding a contradiction.

If a final marking m'_1 could be reached from m' by performing $y \in Y$, then an agreeing m_1 can be reached from m. In the second net, all merged interface places p or their derived p^i and p^o are empty at m'_1, as they are at m_1. Hence, m_1 and m'_1 coincide on the common places and m_1 is final. This is a contradiction, and (w, X, Y) is also a fintree failure of the second net. □

Next, we show how to determine the $\mathcal{F}^+_{fin}$-semantics for the composition of two labeled nets. Here and below, we use $\pi_1(w)$ and $\pi_2(w)$ to denote $w|_{\Sigma_1}$ and $w|_{\Sigma_2}$ for labeled nets N_1 and N_2 with alphabets Σ_1 and Σ_2. For a labeled net N/A with $A \subseteq \Sigma$, $\phi(w)$ denotes $w|_{\Sigma \setminus A}$. We canonically extend the notions of $\pi_i(w)$ and $\phi(w)$ pointwise to sets of traces.

Lemma 4.11 ($\mathcal{F}^+_{fin}$-semantics for labeled net composition). *For composable labeled nets N_1 and N_2, we have*

$$\begin{aligned}\mathcal{F}^+_{fin}(N_1 \| N_2) = \{(w, X_1 \cup X_2, Y_1 \cup Y_2) \mid\ & (\pi_1(w), \pi_1(X_1), \pi_1(Y_1)) \in \mathcal{F}^+_{fin}(N_1),\\ & (\pi_2(w), \pi_2(X_2), \pi_2(Y_2)) \in \mathcal{F}^+_{fin}(N_2)\}\,.\end{aligned}$$

Proof. $\subseteq$: Let $E = N_1 \| N_2$ and let (w, X, Y) be a fintree failure of E. Then there exists a marking m with $m_E \stackrel{w}{\Longrightarrow} m$ according to Definition 4.8. Applying Proposition 2.8, we can project w onto $w_1 = \pi_1(w)$ and $w_2 = \pi_2(w)$ such that $w \in w_1 \| w_2$, $m_{N_1} = m_E|_{P_1} \stackrel{w_1}{\Longrightarrow} m|_{P_1}$, and $m_{N_2} = m_E|_{P_2} \stackrel{w_2}{\Longrightarrow} m|_{P_2}$. For each $x \in X$, we have $m|_{P_1} \stackrel{\pi_1(x)}{\not\Longrightarrow}$ or $m|_{P_2} \stackrel{\pi_2(x)}{\not\Longrightarrow}$ by Proposition 3.11. Set X_1 consists of the x with the first and set X_2 of the x with the second property, where X_1 and X_2 might overlap. Likewise, we can subdivide Y by Proposition 2.8. Note that if a $y \in Y$ can be performed at m at all, then it does not reach a final marking; that is, neither $\pi_1(y)$ nor $\pi_2(y)$ ever reach a final marking. Thus, no trace in $\pi_1(X_1)$ can be performed from $m|_{P_1}$ and no trace in $\pi_1(Y_1)$ can reach a final marking from $m|_{P_1}$. Hence, $(\pi_1(w), \pi_1(X_1), \pi_1(Y_1)) \in \mathcal{F}^+_{fin}(N_1)$, and similarly for N_2.

$\supseteq$: Likewise, given $(w, X_1 \cup X_2, Y_1 \cup Y_2)$ with $(\pi_1(w), \pi_1(X_1), \pi_1(Y_1)) \in \mathcal{F}^+_{fin}(N_1)$ and $(\pi_2(w), \pi_1(X_2), \pi_1(Y_2)) \in \mathcal{F}^+_{fin}(N_2)$ due to m_1 and m_2, one finds that w is a trace of E reaching $m_1 + m_2$, which justifies the first fintree failure. □

From the $\mathcal{F}^+$-semantics in Proposition 3.12, we adapt the following construction of $\mathcal{F}^+_{fin}(N/A)$.

Lemma 4.12. *For any labeled net N and any label set $A \subseteq \Sigma^*_N$, we have*

$$\mathcal{F}^+_{fin}(N/A) = \{(\phi(w), X, Y) \mid (w, \phi^{-1}(X), \phi^{-1}(Y)) \in \mathcal{F}^+_{fin}(N)\}\,.$$

The next proposition yields the $\mathcal{F}^+_{fin}$-semantics for the composition of two open nets.

Proposition 4.13 ($\mathcal{F}^+_{fin}$-semantics for open net composition). *For composable open nets N_1 and N_2, we have*

$$\begin{aligned}\mathcal{F}^+_{fin}(N_1 \oplus N_2) = \{(w, X, Y) \mid \exists (w_1, X_1, Y_1) \in \mathcal{F}^+_{fin}(N_1), (w_2, X_2, Y_2) \in \mathcal{F}^+_{fin}(N_2) :\\ w \in w_1 \Uparrow w_2 \wedge \forall x \in X, y \in Y :\\ (x \in x_1 \Uparrow x_2 \textit{ implies } x_1 \in X_1 \vee x_2 \in X_2)\\ \wedge\, (y \in y_1 \Uparrow y_2 \textit{ implies } y_1 \in Y_1 \vee y_2 \in Y_2)\}\,.\end{aligned}$$

Proof. Follows the same argumentation as the proof of Proposition 3.13, because the sets X and Y are treated analogously. □

Having characterized the $\mathcal{F}^+_{fin}$-semantics for composition and hiding, we shall show that $\mathcal{F}^+_{fin}$-refinement is a precongruence for the composition operator $\oplus$. First, we show the precongruence result for labeled nets and operator $\|$. Then, we show that this result is also preserved under hiding. Finally, we combine these results to show the precongruence for open nets and the operator $\oplus$.

For the first and second step, we can build upon the proof ideas introduced for should testing in [29, Lemma 46], where saturation conditions like SAT1-3 below are employed. The key idea in [29] is to shift traces from the refusal set of *Impl*. We apply the same proof strategy for the X-part of the fintree failures, which is closed under suffix (SAT3). Because this does not hold for the Y-part, we cannot directly apply this idea here. We overcome this problem by adding the set X to the set Y, thereby using the fourth of the following saturation conditions on fintree failures.

SAT1: $(w, X, Y) \in \mathcal{F}^+_{fin}(N), X' \subseteq X, Y' \subseteq Y$ implies $(w, X', Y') \in \mathcal{F}^+_{fin}(N)$
SAT2: $(w, X, Y) \in \mathcal{F}^+_{fin}(N) \wedge \forall z \in Z : (wz, z^{-1}X, z^{-1}Y) \notin \mathcal{F}^+_{fin}(N)$ implies $(w, X \cup Z, Y \cup Z) \in \mathcal{F}^+_{fin}(N)$
SAT3: $(w, X, Y) \in \mathcal{F}^+_{fin}(N)$ implies $(w, \uparrow X, Y) \in \mathcal{F}^+_{fin}(N)$
SAT4: $(w, X, Y) \in \mathcal{F}^+_{fin}(N)$ implies $(w, X, X \cup Y) \in \mathcal{F}^+_{fin}(N)$

SAT1 states that, given a fintree failure (w, X, Y), the sets X and Y can be arbitrarily decreased and the resulting triple is again a fintree failure. Furthermore, the refusal part of $\mathcal{F}^+_{fin}$ is saturated in the sense that the sets X and Y can be extended by any set of traces z such that $(wz, z^{-1}X, z^{-1}Y) \notin \mathcal{F}^+_{fin}(N)$ (SAT2). SAT3 states that the X-part is closed under suffix, and SAT4 shows that the refusal part of $\mathcal{F}^+_{fin}$ is saturated in the sense that the set X can be added to Y.

SAT1, SAT3 and SAT4 are obvious; to see SAT2, assume that some z could be performed from the marking m justifying the tree failure (w, X, Y).

Remark 4.1. We have also explored the idea to encode each $w \in Y$ by $w\checkmark$ for a new symbol $\checkmark$. Then, one can add the resulting traces to X and work with something that looks like an ordinary tree failure. The hope was that this would allow us to use the result [29, Lemma 46] instead of its proof idea, but we have not managed to show the necessary saturation conditions for the domain used in [29].

Lemma 4.14. *$\mathcal{F}^+_{fin}$-refinement is a precongruence for labeled nets for the composition operator $\|$.*

Proof. Let *Impl* and *Spec* be interface-equivalent open nets, and let open net C be composable with *Spec*. Let further $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+} \mathit{Spec}$. We show that $\mathit{Impl}\|C \sqsubseteq_{\mathcal{F}^+} \mathit{Spec}\|C$, following to a large extent the proof of [29, Lemma 46]. For understandability, we also show the full proof for the case $\Sigma_{Spec} = \Sigma_C$ here, because in this case the projection functions in the construction of the synchronized fintree failures become the identity over the complete alphabet and hence disappear.

Consider a fintree failure $(w, X_{Impl} \cup X_C, Y_{Impl} \cup Y_C) \in \mathcal{F}^+_{fin}(\mathit{Impl}\|C)$ such that $(w, X_{Impl}, Y_{Impl}) \in \mathcal{F}^+_{fin}(\mathit{Impl})$ and $(w, X_C, Y_C) \in \mathcal{F}^+_{fin}(C)$. Define the set

$$W = \{v \mid (wv, v^{-1}X_C, v^{-1}Y_C) \notin \mathcal{F}^+_{fin}(C)\}$$

which contains those traces that can be added to X_C and Y_C according to SAT2. We shift the traces in W from *Impl* to C. To this end, we define four sets

$$\begin{aligned} X'_{Impl} &= X_{Impl} \setminus \uparrow W, \\ Y'_{Impl} &= Y_{Impl} \setminus \uparrow W, \\ X'_C &= X_C \cup \uparrow W, \\ Y'_C &= Y_C \cup \uparrow W\,. \end{aligned}$$

We immediately see: $X_{Impl} \cup X_C \subseteq X'_{Impl} \cup X'_C$ as well as $Y_{Impl} \cup Y_C \subseteq Y'_{Impl} \cup Y'_C$ and $X'_{Impl} \cup Y'_{Impl} \subseteq X_{Impl} \cup Y_{Impl} \cup X_C \cup Y_C$. By SAT1, we have $(w, X'_{Impl}, Y'_{Impl}) \in \mathcal{F}^+_{fin}(\mathit{Impl})$.

Due to $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+} \mathit{Spec}$, there exists $x \in \{\varepsilon\} \cup \downarrow X'_{Impl} \cup \downarrow Y'_{Impl}$ such that

$$(wx, x^{-1}X'_{Impl}, x^{-1}Y'_{Impl}) \in \mathcal{F}^+_{fin}(\mathit{Spec}) \qquad (1).$$

We have $x \notin \uparrow W$. Assume the contrary: $x = \varepsilon$ implies $\varepsilon \in W$ which is a contradiction to the construction of W. $x \in \downarrow X'_{Impl}$ implies $\exists x' \in X'_{Impl} : x \sqsubseteq x' \wedge \exists v \in W : v \sqsubseteq x \sqsubseteq x'$ which is a contradiction to the definition of X'_{Impl}. The same argument also applies to $x \in Y'_{Impl}$.

From $x \notin W$, it follows that $(wx, x^{-1}X_C, x^{-1}Y_C) \in \mathcal{F}^+_{fin}(C)$. Further, for all $u \in x^{-1}W$ (i.e., $xu \in W$), $(wxu, u^{-1}x^{-1}X_C, u^{-1}x^{-1}Y_C) \notin \mathcal{F}^+_{fin}(C)$.

By SAT2, $(wx, x^{-1}(X_C \cup W), x^{-1}(Y_C \cup W)) \in \mathcal{F}^+_{fin}(C)$. Consider now the second ingredient, $x^{-1}(X_C \cup W)$. By SAT3, this implies $\uparrow x^{-1}(X_C \cup W)$. With Lemma 2.14(3), we have $\uparrow x^{-1}(X_C \cup W) \supseteq x^{-1}(X_C \cup \uparrow W) = x^{-1}X'_C$. Now, according to SAT1, $x^{-1}X'_C$ can replace $x^{-1}(X_C \cup W)$.

Consider now the third ingredient, $x^{-1}(Y_C \cup W)$. By SAT4, we extend this set to $x^{-1}(Y_C \cup W) \cup x^{-1}X'_C \supseteq x^{-1}{\uparrow} W \cup x^{-1}Y_C = x^{-1}({\uparrow} W \cup Y_C) = x^{-1}Y'_C$. Now, SAT1 allows that $x^{-1}Y'_C$ can replace $x^{-1}(Y_C \cup W)$.

Combining these results, we get $(wx, x^{-1}X'_C, x^{-1}Y'_C) \in \mathcal{F}^+_{fin}(C)$. Then, by Lemma 4.11 and (1), we obtain $(wx, x^{-1}(X'_{Impl} \cup X'_C), x^{-1}(Y'_{Impl} \cup Y'_C)) \in \mathcal{F}^+_{fin}(Spec \| C)$. By SAT1, we have $(wx, x^{-1}(X_{Impl} \cup X_C), x^{-1}(Y_{Impl} \cup Y_C)) \in \mathcal{F}^+_{fin}(Spec \| C)$ where $x \in (\{\varepsilon\} \cup {\downarrow} X'_{Impl} \cup {\downarrow} Y'_{Impl}) \subseteq (\{\varepsilon\} \cup {\downarrow} X_{Impl} \cup {\downarrow} Y_{Impl} \cup {\downarrow} X_C \cup {\downarrow} Y_C)$.

Now consider the general case. Let π and π_C denote projections, projecting onto the alphabets $\Sigma_{Spec} = \Sigma_{Impl}$ and Σ_C, respectively. We have

1. $\pi(V \cup W) = \pi(V) \cup \pi(W)$
2. $\pi({\uparrow} V) \subseteq {\uparrow} \pi(V)$
3. $\pi(w^{-1}V) \subseteq \pi(w)^{-1}\pi(V)$

Consider a fintree failure $(w, X_{Impl} \cup X_C, Y_{Impl} \cup Y_C) \in \mathcal{F}^+_{fin}(Impl \| C)$ such that $(\pi(w), \pi(X_{Impl}), \pi(Y_{Impl})) \in \mathcal{F}^+_{fin}(Impl)$ and $(\pi_C(w), \pi_C(X_C), \pi_C(Y_C)) \in \mathcal{F}^+_{fin}(C)$. Define the set

$$W = \{v \mid (\pi_C(wv), \pi_C(v^{-1}X_C), \pi_C(v^{-1}Y_C)) \notin \mathcal{F}^+_{fin}(C)\}\,.$$

We shift the traces in W from $Impl$ to C. To this end, we define four sets

$$\begin{aligned} X'_{Impl} &= X_{Impl} \setminus {\uparrow} W, \\ Y'_{Impl} &= Y_{Impl} \setminus {\uparrow} W, \\ X'_C &= X_C \cup {\uparrow} W, \\ Y'_C &= Y_C \cup {\uparrow} W\,. \end{aligned}$$

We immediately see: $X_{Impl} \cup X_C \subseteq X'_{Impl} \cup X'_C$ as well as $Y_{Impl} \cup Y_C \subseteq Y'_{Impl} \cup Y'_C$ and $X'_{Impl} \cup Y'_{Impl} \subseteq X_{Impl} \cup Y_{Impl} \cup X_C \cup Y_C$. By SAT1, we have $(\pi(w), \pi(X'_{Impl}), \pi(Y'_{Impl})) \in \mathcal{F}^+_{fin}(Impl)$.

Because of $Impl \sqsubseteq_{\mathcal{F}^+} Spec$, there exists $x \in \{\varepsilon\} \cup {\downarrow} X'_{Impl} \cup {\downarrow} Y'_{Impl}$ such that $(\pi(wx), \pi(x)^{-1}\pi(X'_{Impl}), \pi(x)^{-1}\pi(Y'_{Impl})) \in \mathcal{F}^+_{fin}(Spec)$. Hence, we have

$$(\pi(wx), \pi(x^{-1}X'_{Impl}), \pi(x^{-1}Y'_{Impl})) \in \mathcal{F}^+_{fin}(Spec) \qquad (2)$$

due to Item 3 and SAT1.

Again, trace $x \notin {\uparrow} W$ (by the same argumentation as in the proof of the case $\Sigma_{Spec} = \Sigma_C$), and we conclude that $(\pi_C(wx), \pi_C(x^{-1}X_C), \pi_C(x^{-1}Y_C)) \in \mathcal{F}^+_{fin}(C)$. Further, for all $u \in x^{-1}W$ (i.e., $xu \in W$), $(\pi_C(wxu), \pi_C(u^{-1}x^{-1}X_C), \pi_C(u^{-1}x^{-1}Y_C)) \notin \mathcal{F}^+_{fin}(C)$, due to the definition of W.

Now, $(\pi_C(wxu), \pi_C(u)^{-1}\pi_C(x^{-1}X_C), \pi_C(u)^{-1}\pi_C(x^{-1}Y_C)) \notin \mathcal{F}^+_{fin}(C)$ with Item 3 and SAT1, and we obtain, by Item 1 and SAT2,

$$(\pi_C(wx), \pi_C(x^{-1}(X_C \cup W)), \pi_C(x^{-1}(Y_C \cup W))) \in \mathcal{F}^+_{fin}(C)\,.$$

Consider the second ingredient, $\pi_C(x^{-1}(X_C \cup W))$ of this fintree failure. Applying SAT3, we obtain $\uparrow \pi_C(x^{-1}(X_C \cup W))$ and with SAT1 and Item 2, $\pi_C(\uparrow x^{-1}(X_C \cup W))$. Because $x \notin \uparrow W$, we can apply Lemma 2.14(3) and SAT1, and we arrive at $\pi_C(x^{-1}(X_C \cup \uparrow W)) = \pi_C(x^{-1}X'_C)$.

For the third ingredient $\pi_C(x^{-1}(Y_C \cup W))$ of this fintree failure, we obtain by SAT4 $\pi_C(x^{-1}(Y_C \cup W)) \cup \pi_C(x^{-1}X'_C)$. By Item 1, we can transform this into $\pi_C(x^{-1}(Y_C \cup W \cup X'_C))$ and by SAT1 into $\pi_C(x^{-1}(Y_C \cup \uparrow W)) = \pi_C(x^{-1}Y'_C)$.

Combining these results yields $(\pi_C(wx), \pi_C(x^{-1}X'_C), \pi_C(x^{-1}Y'_C)) \in \mathcal{F}^+_{fin}(C)$. Then, with Lemma 4.11 and (2), we obtain

$$(wx, x^{-1}(X'_{Impl} \cup X'_C), x^{-1}(Y'_{Impl} \cup Y'_C)) \in \mathcal{F}^+_{fin}(Spec \| C)\,.$$

Applying SAT1 yields $(wx, x^{-1}(X_{Impl} \cup X_C), x^{-1}(Y_{Impl} \cup Y_C) \in \mathcal{F}^+_{fin}(Spec \| C)$ where $x \in (\{\varepsilon\} \cup\downarrow X'_{Impl} \cup\downarrow Y'_{Impl}) \subseteq (\{\varepsilon\} \cup\downarrow X_{Impl} \cup\downarrow Y_{Impl} \cup\downarrow X_C \cup\downarrow Y_C)$. □

Remark 4.2. The proof of Lemma 4.14 is not restricted to sets of fintree failures of labeled nets, but holds for general sets of fintree failures for which the four saturation SAT1 – SAT4 hold.

We show that $\mathcal{F}^+_{fin}$-refinement for labeled nets is preserved under hiding.

Lemma 4.15. *$\mathcal{F}^+_{fin}$-refinement for labeled nets is preserved under hiding.*

Proof. Let *Impl* and *Spec* be interface-equivalent labeled nets such that *Impl* $\sqsubseteq_{\mathcal{F}^+_{fin}}$ *Spec*. Further, let $A \subseteq \Sigma^*$ and $(w, X, Y) \in \mathcal{F}^+_{fin}(Impl/A)$. Consider $(v, \phi^{-1}(X), \phi^{-1}(Y)) \in \mathcal{F}^+_{fin}(Impl)$ with $w = \phi(v)$. Because *Impl* $\mathcal{F}^+_{fin}$-refines *Spec*, there is $x \in \{\varepsilon\} \cup\downarrow \phi^{-1}(X) \cup\downarrow \phi^{-1}(Y)$ with $(vx, x^{-1}\phi^{-1}(X), x^{-1}\phi^{-1}(Y)) \in \mathcal{F}^+_{fin}(Spec)$. It can be shown that $\phi^{-1}(\phi(x)^{-1}X) = x^{-1}\phi^{-1}(X)$. Using this observation together with $(v, \phi^{-1}(X), \phi^{-1}(Y)) \in \mathcal{F}^+_{fin}(Impl)$, we conclude that $(\phi(vx), \phi(x)^{-1}X, \phi(x)^{-1}Y) \in \mathcal{F}^+_{fin}(Spec/A)$ and additionally $(\phi(v)\phi(x), \phi(x)^{-1}X, \phi(x)^{-1}Y) \in \mathcal{F}^+_{fin}(Spec/A)$. Because $\phi(v) = w$ and $\phi(x) \in \{\varepsilon\} \cup \downarrow(X) \cup \downarrow(Y)$, the lemma holds. □

With Lemma 4.14 and Lemma 4.15, we have the ingredients to show that $\mathcal{F}^+_{fin}$-refinement is also a precongruence for the composition operator $\oplus$.

Theorem 4.16 (precongruence). *$\mathcal{F}^+_{fin}$-refinement is a precongruence for the composition operator $\oplus$.*

Proof. The proof is analogous to the proof of Theorem 3.15. By Lemma 4.14 and Lemma 4.15, $\mathcal{F}^+_{fin}$-refinement is a precongruence for the composition operator $\Uparrow$ and, by Lemma 4.10, also for for the composition operator $\oplus$. □

With the next theorem, we show that $\mathcal{F}^+_{fin}$-refinement and the coarsest precongruence that is contained in the *fr*-accordance relation coincide.

Theorem 4.17 (precongruence and $\mathcal{F}^+_{fin}$-refinement coincide). *For two interface-equivalent open nets Impl and Spec, we have*

$$\mathit{Impl} \sqsubseteq_{\mathcal{F}^+_{fin}} \mathit{Spec} \quad \mathit{iff} \quad \mathit{Impl} \sqsubseteq^c_{fr,acc} \mathit{Spec}\,.$$

Proof. $\Rightarrow$: Consider a trace $w \in \mathit{stop}(\mathit{Impl})$ ($w \in \mathit{dead}(\mathit{Impl})$); we prove $w \in \mathit{stop}(\mathit{Spec})$ ($w \in \mathit{dead}(\mathit{Spec})$). Then, applying Theorem 4.7, we get $\mathit{Impl} \sqsubseteq_{fr,acc} \mathit{Spec}$, and this in turn also shows the claim with Theorem 4.16 and the definition of $\sqsubseteq^c_{fr,acc}$. So let O be the set of output places of *Impl* and of *Spec*.

We have $w \in \mathit{stop}(\mathit{Impl})$ iff $(w, O, \emptyset) \in \mathcal{F}^+_{fin}(\mathit{Impl})$ by Definitions 3.4 and 3.9. Then, by $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+_{fin}} \mathit{Spec}$, there must be a suitable $x \in \{\varepsilon\} \cup O = \{\varepsilon\} \cup {\downarrow} O$ that satisfies the defining condition of Definition 4.8. We cannot have $x \in O$ because $(wx, \{\varepsilon\}, \emptyset) \notin \mathcal{F}^+_{fin}(\mathit{Spec})$ by Definition 4.8. Thus, $x = \varepsilon$ and $(w, O, \emptyset) \in \mathcal{F}^+_{fin}(\mathit{Spec})$, implying $w \in \mathit{stop}(\mathit{Spec})$.

We have $w \in \mathit{dead}(\mathit{Impl})$ iff $(w, O, \{\varepsilon\}) \in \mathcal{F}^+_{fin}(\mathit{Impl})$ by Definition 4.3. Again, $x = \varepsilon$ and thus $(w, O, \{\varepsilon\}) \in \mathcal{F}^+_{fin}(\mathit{Spec})$, implying $w \in \mathit{dead}(\mathit{Spec})$.

$\Leftarrow$: Suppose $\mathit{Impl} \sqsubseteq^c_{fr,acc} \mathit{Spec}$, and let $(w, X, Y) \in \mathcal{F}^+_{fin}(\mathit{Impl})$. In addition, consider an open net C with the new output x and the new input y. Open net C has the empty initial marking and contains only a single transition that can indefinitely repeat to produce a token in x while consuming a token from place y. In addition, its final marking is the empty marking. The idea is to construct an open net N from (w, X, Y) such that C is not an *fr*-controller of $\mathit{Impl} \oplus N$ because of (w, X, Y). By $\mathit{Impl} \sqsubseteq^c_{fr,acc} \mathit{Spec}$ and because $\sqsubseteq^c_{fr,acc}$ is a precongruence, we have $\mathit{Impl} \oplus N \sqsubseteq^c_{fr,acc} \mathit{Spec} \oplus N$ and thus $\mathit{Impl} \oplus N \sqsubseteq_{fr,acc} \mathit{Spec} \oplus N$ by Definition 4.2. Hence, C is also not an *fr*-controller of $\mathit{Spec} \oplus N$, and from this we shall conclude that (w, X, Y) is covered by $\mathcal{F}^+_{fin}(\mathit{Spec})$ according to Definition 4.9. Then we will have proved $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+_{fin}} \mathit{Spec}$.

The construction of the open net N is similar to the one in the proof of (the reverse implication of) Theorem 3.17; the tree part on the right-hand side of Fig. 6 corresponds to $X \cup Y$ now. The open net N has input places $I = O_{Impl} \uplus \{x\}$, output places $O = I_{Impl} \uplus \{y\}$, and enables a transition sequence $v = t_1 \dots t_k$. Each transition in v is connected to an interface place of N such that the corresponding trace of interface actions is w; that is, the net N contains net N_w as in Fig. 5. Thus, we can essentially fire the trace w of $\mathit{env}(N)$ in $\mathit{Impl} \oplus N$ and, therefore, in $\mathit{Impl} \oplus N \oplus C$ by firing v instead of the labeled transitions. This way, we reach in *Impl* a marking m that refuses X in $\mathit{env}(\mathit{Impl})$; in N, there is only one token in a place p_ε and the token in a place p has been consumed. This token is necessary to enable transition t' that is—together with transition t—essential for f-responsiveness, because they can repeatedly communicate with C. The place p can only be marked again by firing some transition t'_x with $x \in X$, and this in turn requires the firing of a transition sequence that—similarly to v—looks to *Impl* like the trace x. But this trace cannot be fired at m. In addition, every trace $y \in Y$ that cannot lead to a final marking in *Impl* leads to a final marking in the tree part of N. This construction guarantees that there is a marking reachable in the composition $\mathit{Impl} \oplus N \oplus C$ which is neither responsive

(because place p is not marked and hence there is no communication between C and N) nor reaches a final marking (because if $N \oplus C$ is in a final marking, then *Impl* is not). As a consequence, $\mathit{Impl} \oplus N \oplus C$ is not f-responsive and, thus, C is not an *fr*-controller of $\mathit{Impl} \oplus N$.

To achieve the effect just described, the second part of the open net N encodes the tree part for X and Y of fintree failure (w, X, Y); this second part is a tree representing $X \cup Y$. Common prefixes thereby correspond to the same path in this part. If a path corresponds to some $y \in Y$, a token on the place at the end of this path is a final marking of N; if for example $b \in Y$, then the marking with just one token on the place p_b is final; see Fig. 6. For a path corresponding to some $x \in X$, a token in the respective place allows to mark p again.

Let $w = w_1 \dots w_k$ such that for $j = 1, \dots, k$, $w_j \in I_{Impl} \uplus O_{Impl}$. Let $N = (P, T, F, m_N, O, I, \Omega)$ be an open net like the one in the proof of Theorem 3.17, but replace every occurrence of $\downarrow X$ with $\downarrow X \cup \downarrow Y$ and add $\Omega = \{[p_z] \mid z \in Y\}$.

As argued previously, we now have that C is not an *fr*-controller of $\mathit{Spec} \oplus N$; that is, some marking m_1 can be reached in $\mathit{Spec} \oplus N \oplus C$ where f-responsiveness is violated. Clearly, places p, x, and y must be empty in m_1; thus, v has been fired in N plus possibly some transitions in the fintree part of the net. There is just one token in the places of $\mathit{inner}(N)$, and it is in some p_u with $uu' \in X$ (resp. $uu' \in Y$). Let m_2 be the projection of m_1 onto the places of *Spec*. From the point of view of *Spec*, we have fired a trace wu of $\mathit{env}(\mathit{Spec})$ reaching m_2. Because in $\mathit{Spec} \oplus N \oplus C$ no $t'_{uu'}$ can become enabled and the composition cannot reach a final marking—otherwise, C would be an *fr*-controller—no u' can be fired in $\mathit{env}(\mathit{Spec})$ at m_2 and a final marking is not reachable. Thus, we conclude $(wu, \{u' \mid uu' \in X\}, \{u' \mid uu' \in Y\}) \in \mathcal{F}^+_{\mathit{fin}}(\mathit{Spec})$ and, therefore, $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+_{\mathit{fin}}} \mathit{Spec}$. □

Example 4.7. As shown in Example 4.1, $S' \sqsubseteq^c_{\mathit{fr},\mathit{acc}} S$ does not hold. This is now confirmed with Theorem 4.17, because S' does not $\mathcal{F}^+_{\mathit{fin}}$-refine S by Example 4.6.

5 Bounded nets without responsiveness

In the two previous sections, we presented precongruences for responsiveness and f-responsiveness. The former notion does not take into account final markings whereas the latter notion does. In this section, we study another property of open net compositions: *boundedness*. We first consider boundedness in isolation—that is, without requiring open nets to be responsive or to have final markings—and add boundedness to responsiveness and f-responsiveness in the subsequent sections. That way, technicalities become simpler.

The next two definitions define the notion of a controller and of accordance for boundedness.

Definition 5.1 (b-controller, b-accordance). An open net C is a *b-controller* of an open net N if their composition $N \oplus C$ is a b-bounded, closed net.

For interface-equivalent open nets *Impl* and *Spec*, *Impl b-accords with Spec*, denoted by *Impl* $\sqsubseteq_{b,acc}$ *Spec*, if for all open nets C: C is a b-controller of *Spec* implies C is a b-controller of *Impl*.

Example 5.1. The open net C' in Fig. 7a is a 1-controller of the open net S in Fig. 1a because their composition $S \oplus C'$ in Fig. 7b is a 1-bounded closed net. Consequently, C' is a b-controller of S for every $b \in \mathbb{N}_+$. In contrast, the open net C in Fig. 1b is not a b-controller of S for any $b \in \mathbb{N}_+$ because the place r is unbounded in their composition $S \oplus C$ in Fig. 1c after the error has been caught.

While every open net has at least one r-controller and one fr-controller, there exist open nets that do not have any b-controller. An example is an open net that performs a self loop and in every cycle produces a token in an output place, thereby violating any bound.

In the rest of this section, we give a trace-based semantics for open nets based on which we prove b-accordance to be a precongruence with regard to the open net composition operator $\oplus$.

Our trace-based semantics for b-boundedness of an open net N is part of the b-bounded *stopdead*-semantics [31,32]. A bound violation is a marking that is not b-bounded, and we investigate the traces leading to such a bound violation, called *strict bound$_b$-violators*. A bound violation is regarded as catastrophic because it cannot be corrected. Thus, the behavior after a bound violation does not matter, and we will hide all possible differences by treating all strict *bound$_b$*-violators and their continuations in the same way. Technically, we achieve the hiding by including all continuations of strict *bound$_b$*-violators in a set *bound$_b$*, the set of *bound$_b$-violators*. For the same reason, *bound$_b$* is contained in the second component of our *b-bounded semantics*, the language of N. This technique is called *flooding* in [14].

Definition 5.2 (b-bounded semantics). Let N be a labeled net. A trace w is a *strict bound$_b$-violator* of N if there exists a marking m with $m_N \stackrel{w}{\Longrightarrow} m$ that is not b-bounded; a continuation of a strict *bound$_b$*-violator of N is a *bound$_b$-violator* of N. The *b-bounded semantics* of N is defined by

- $bound_b(N) = \{w \in (I \uplus O)^* \mid w \text{ is a bound}_b\text{-violator of } N\}$ and
- $L_b(N) = L(N) \cup bound_b(N)$.

We recall properties of *bound$_b$* and L_b, and how the *bound$_b$*- and L_b-semantics of a composition is calculated. We also consider a labeled net N/A, $A \subseteq \Sigma$ and we use $\phi(w)$ to denote $w|_{\Sigma \setminus A}$. We canonically extend the notion of $\phi(w)$ pointwise to sets of traces.

Proposition 5.3. *For two composable labeled nets N_1 and N_2, we have*

1. $bound_b(N_1 \| N_2) = \uparrow \Big(\big(bound_b(N_1) \| L_b(N_2)\big) \cup \big(L_b(N_1) \| bound_b(N_2)\big) \Big)$
2. $bound_b(N_1 \Uparrow N_2) = \uparrow \Big(\big(bound_b(N_1) \Uparrow L_b(N_2)\big) \cup \big(L_b(N_1) \Uparrow bound_b(N_2)\big) \Big)$,

3. $L_b(N_1 \| N_2) = (L_b(N_1) \| L_b(N_2)) \cup bound_b(N_1 \| N_2)$,
4. $L_b(N_1 \Uparrow N_2) = (L_b(N_1) \Uparrow L_b(N_2)) \cup bound_b(N_1 \Uparrow N_2)$,
5. $bound_b(N/A) = \{\phi(w) \mid w \in bound_b(N)\}$,
6. $L_b(N/A) = \{\phi(w) \mid w \in L_b(N)\}$,

and for two composable open nets N_1 and N_2, we have

7. $bound_b(N_1 \oplus N_2) = \uparrow \Big(\big(bound_b(N_1) \Uparrow L_b(N_2) \big) \cup \big(L_b(N_1) \Uparrow bound_b(N_2) \big) \Big)$
8. $L_b(N_1 \oplus N_2) = \big(L_b(N_1) \Uparrow L_b(N_2) \big) \cup bound_b(N_1 \oplus N_2)$.

Proof. (1) has already been proved for $b = 1$ in [33, Theorem 3.3.3]; we can use the same considerations to show that this result can be generalized to an arbitrary bound $b \in \mathbb{N}_+$; (2) follows directly from (1) by Definition 2.7;

(3)
$$\begin{aligned}
& L_b(N_1 \| N_2) \\
= \;& (L(N_1) \cup bound_b(N_1)) \| (L(N_2) \cup bound_b(N_2)) \cup bound_b(N_1 \| N_2) \\
= \;& (L(N_1) \| L(N_2)) \cup (bound_b(N_1) \| L(N_2)) \cup (bound_b(N_2) \| L(N_1)) \\
& \cup (bound_b(N_1) \| bound_b(N_2)) \cup bound_b(N_1 \| N_2) \\
& \quad \{ \text{ By } L(N_2) \subseteq L_b(N_2), L(N_1) \subseteq L_b(N_1), bound_b(N_1) \subseteq L_b(N_1), (1) \} \\
= \;& L(N_1) \| L(N_2) \cup bound_b(N_1 \| N_2) \\
& \quad \{ \text{ by [33, Theorem 3.1.7(4)] } \} \\
= \;& L(N_1 \| N_2) \cup bound_b(N_1 \| N_2)
\end{aligned}$$

(4) follows the same argumentation as (3); (5) and (6) are obvious; and (7) and (8) have already been proved in [32, Theorem 30]. □

If we consider the composition of two open nets N_1 and N_2, then its b-bounded semantics coincides with that of the parallel composition of the two environments $env(N_1) \Uparrow env(N_2)$.

Lemma 5.4. *For two composable open nets N_1 and N_2, we have*

1. $bound_b(N_1 \oplus N_2) = bound_b(env(N_1) \Uparrow env(N_2))$ *, and*
2. $L_b(N_1 \oplus N_2) = L_b(env(N_1) \Uparrow env(N_2))$.

Proof. The first item follows by Proposition 5.3(2) and (7). For the second item, we have $L_b(N_1 \oplus N_2) = L(N_1 \oplus N_2) \cup bound_b(N_1 \oplus N_2)$ by Definition 5.2. With $L(N_1 \oplus N_2) = L(env(N_1) \Uparrow env(N_2))$ by [32, Theorem 18(1)] and the first item, we conclude with Definition 5.2 that $L_b(N_1 \oplus N_2) = L_b(env(N_1) \Uparrow env(N_2))$. □

The presence of $bound_b$-violators of N_1 and N_2 is closely related to the question whether N_1 and N_2 are b-controller of each other. This result directly follows from Proposition 5.3(7).

Corollary 5.5 (b-boundedness vs. b-bounded semantics). *For two composable open nets N_1 and N_2 such that $N_1 \oplus N_2$ is a closed net, we have*

$$N_1 \text{ is a } b\text{-controller of } N_2 \quad \text{iff} \quad \begin{array}{l} L_b(N_1) \cap bound_b(N_2) = \emptyset \text{ and} \\ bound_b(N_1) \cap L_b(N_2) = \emptyset \,. \end{array}$$

The inclusion of the b-bounded semantics of open nets defines a refinement relation. With the next theorem, we prove that it coincides with b-accordance; in other words, we provide a trace-based characterization for the latter.

Theorem 5.6 (b-accordance and b-bounded semantics inclusion coincide). *For two interface-equivalent open nets Impl and Spec, we have*

$$\mathit{Impl} \sqsubseteq_{b,acc} \mathit{Spec} \quad \text{iff} \quad \begin{array}{l} \mathit{bound}_b(\mathit{Impl}) \subseteq \mathit{bound}_b(\mathit{Spec}) \text{ and} \\ L_b(\mathit{Impl}) \subseteq L_b(\mathit{Spec})\,. \end{array}$$

Proof. $\Rightarrow$: Use the proof of [32, Theorem 33].

$\Leftarrow$: Proof by contraposition. Consider an open net C such that $\mathit{Impl} \oplus C$ and $\mathit{Spec} \oplus C$ are closed nets. Assume that C is not a b-controller of Impl. Then we have $L_b(\mathit{Impl}) \cap \mathit{bound}_b(C) \neq \emptyset$ or $\mathit{bound}_b(\mathit{Impl}) \cap L_b(C) \neq \emptyset$ by Corollary 5.5. Because of the assumed inclusions, we also have $L_b(\mathit{Spec}) \cap \mathit{bound}_b(C) \neq \emptyset$ or $\mathit{bound}_b(\mathit{Spec}) \cap L_b(C) \neq \emptyset$. Again with Corollary 5.5, we see that C is not a b-controller of Spec. □

We show that b-accordance is a precongruence for the composition operators $\|$ and $\oplus$.

Proposition 5.7. *Inclusion of bound_b- and L_b-traces is a precongruence for $\|$ and hiding on labeled nets.*

Proof. Follows from Proposition 5.3(1),(3),(5),(6).

Theorem 5.8 (b-accordance is a precongruence). *The b-accordance relation $\sqsubseteq_{b,acc}$ is a precongruence w.r.t. the composition operator $\oplus$.*

Proof. Let *Impl* and *Spec* be interface-equivalent open nets such that $\mathit{Impl} \sqsubseteq_{b,acc} \mathit{Spec}$, and let C be an open net that is composable with *Impl* (and *Spec* by interface equivalence). Then:

$$\begin{aligned}
\mathit{bound}_b(\mathit{Impl} \oplus C) &= \{ \text{ by Proposition 5.3(7) } \} \\
&\quad \uparrow \Big(\big(\mathit{bound}_b(\mathit{Impl}) \Uparrow L_b(C)\big) \cup \big(L_b(\mathit{Impl}) \Uparrow \mathit{bound}_b(C)\big) \Big) \\
&\subseteq \{ \text{ by Theorem 5.6 and Definition 2.7 } \} \\
&\quad \uparrow \Big(\big(\mathit{bound}_b(\mathit{Spec}) \Uparrow L_b(C)\big) \cup \big(L_b(\mathit{Spec}) \Uparrow \mathit{bound}_b(C)\big) \Big) \\
&= \mathit{bound}_b(\mathit{Spec} \oplus C) \text{ by Proposition 5.3(7)} \\
L_b(\mathit{Impl} \oplus C) &= \{ \text{ by Proposition 5.3(8) } \} \\
&\quad \big(L_b(\mathit{Impl}) \Uparrow L_b(C)\big) \cup \mathit{bound}_b(\mathit{Impl} \oplus C) \\
&\subseteq \{ \text{ by Theorem 5.6 and Definition 2.7 } \} \\
&\quad \big(L_b(\mathit{Spec}) \Uparrow L_b(C)\big) \cup \mathit{bound}_b(\mathit{Impl} \oplus C) \\
&\subseteq \{ \text{ by } \mathit{bound}_b(\mathit{Impl} \oplus C) \subseteq \mathit{bound}_b(\mathit{Spec} \oplus C)\} \\
&\quad \big(L_b(\mathit{Spec}) \Uparrow L_b(C)\big) \cup \mathit{bound}_b(\mathit{Spec} \oplus C) \\
&= L_b(\mathit{Spec} \oplus C) \text{ by Proposition 5.3(8).}
\end{aligned}$$

Thus, $\mathit{Impl} \oplus C \sqsubseteq_{b,acc} \mathit{Spec} \oplus C$ by Theorem 5.6. □

6 Bounded nets and no final markings

In this section, we require the composition of open nets to be bounded and responsive; that is, we ignore final markings as in Sect. 3. The resulting variant of responsiveness, *bounded responsiveness* or b-responsiveness for short, is similar to safe P-deadlock equivalence in [33].

Definition 6.1 (b-responsiveness). Let N_1 and N_2 be composable open nets. A marking of $N_1 \oplus N_2$ is *b-responsive* if it is responsive and b-bounded. The open nets N_1 and N_2 are b-responsive if their composition $N_1 \oplus N_2$ is a closed net and every reachable marking in $N_1 \oplus N_2$ is *b-responsive*.

Two open nets are b-responsive if at least one net can repeatedly talk while respecting the message bound b. In fact, we can prove that, due to b-responsiveness, each net always has the chance to send a message (possibly after some messages from the other net). Thus, the word 'responsive' is really justified here.

Proposition 6.2. *Let open nets N_1 and N_2 be b-responsive. Then, from any reachable marking m of $N_1 \oplus N_2$, markings m_1 and m_2 are reachable such that $m_1 \xrightarrow{t_1}$ and $m_2 \xrightarrow{t_2}$ with $t_1^\bullet \cap O_1 \neq \emptyset$ and $t_2^\bullet \cap O_2 \neq \emptyset$.*

Proof. Proof by contradiction. Assume that there exists a marking m from which no suitable m_1 or m_2 is reachable. This contradicts Definition 6.1, because all markings in $N_1 \oplus N_2$ are b-responsive and thus responsive.

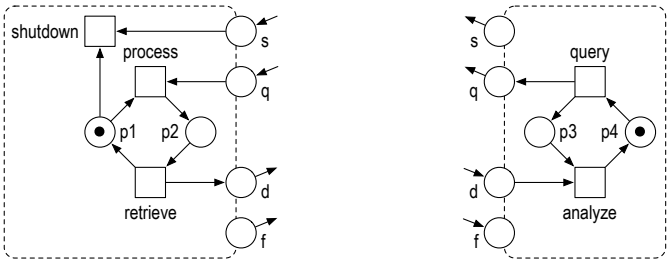
Now assume that there exists an m from which only a marking m_1 is reachable but no m_2. Then, in $N_1 \oplus N_2$ there exists a run to a marking m_1 enabling some t_1. No tokens are put onto $I_1 = O_2$ in this run; otherwise, we would have found an m_2 just before such a firing. Hence, no transitions of N_2 are needed to enable t_1, and we can assume that all transitions of the run belong to N_1. Consequently, no token is removed from $O_1 = I_2$. Now we fire t_1 and reach some m' with at least one token more on O_1. If m' has an m_2' as claimed in the lemma, this can also serve for m as m_2. Hence, m_2' does not exist, but some m_1' must, as argued previously. We repeat this argument, and each time the token count on O_1 increases until bound b is violated. However, this contradicts Definition 6.1, stating that $N_1 \oplus N_2$ is b-bounded. As a consequence, a marking m_2 must be reachable from m. □

We redefine the notion of a controller and of accordance for this variant of responsiveness to br-controller and br-accordance. As br-accordance shall turn out not to be a precongruence, we also introduce its coarsest precongruence.

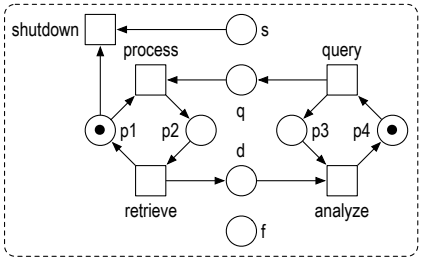
Definition 6.3 (br-controller, br-accordance). An open net C is a *br-controller* of an open net N if N and C are b-responsive.

For interface-equivalent open nets *Impl* and *Spec*, *Impl br-accords with Spec*, denoted by $\mathit{Impl} \sqsubseteq_{br,acc} \mathit{Spec}$, if for all open nets C holds: C is a br-controller of *Spec* implies C is a br-controller of *Impl*.

We denote the coarsest precongruence contained in $\sqsubseteq_{br,acc}$ by $\sqsubseteq^c_{br,acc}$.



(a) Open net D (b) Open net U



(c) Open net $D \oplus U$

Fig. 8: Open nets modeling a database server, a user, and their composition. In addition to the models, we have $\Omega_D = \{[\,]\}$, $\Omega_U = \{[p_4]\}$ and $\Omega_{D \oplus U} = \{[p_4]\}$.

Example 6.1. Figure 8 shows three open systems, each modeled as an open net. The open net D models a database server. After processing a query (input place q), it responds with the retrieved data (output place d). A user may shut down D by sending a shutdown message (input place s). D has the (unused) capability to forward messages (output place f). The open net U models a user of the database. It repeatedly queries the database and analyzes the returned data. U never sends a shutdown message and ignores any forwarded message from D. The open nets D and U are composable. Their composition $D \oplus U$ is a 1-bounded closed net, which is depicted in Fig. 8c.

Figure 9 depicts a modified database server D'. It has the same functionality as D but forwards a shutdown message to the output place f. No br-controller of D sends a message s, as otherwise D could fire *shutdown* and then could not produce any output, contradicting Proposition 6.2. Thus, D' br-accords with D. Vice versa, no br-controller of D' sends a message s because after sending f, D' cannot produce any token on d or f. Thus, also D br-accords with D'. Observe that these two statements hold for any bound b.

Extending the example with the open nets X and Y in Fig. 10, we can show that br-accordance is not compositional: X is a br-controller of $D' \oplus Y$ but not of $D \oplus Y$. Whereas the transition *activate* of Y can be fired in $(D' \oplus Y) \oplus X$ (enabling br-responsiveness), it cannot be fired in $(D \oplus Y) \oplus X$.

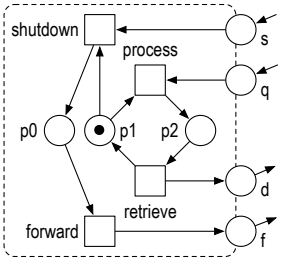
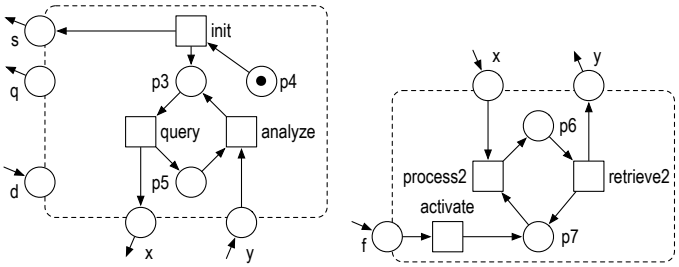


Fig. 9: The open net D' modeling a modified database server. We have $\Omega_{D'} = \{[p_0]\}$.



(a) Open net X (b) Open net Y

Fig. 10: Two open nets proving that *br*-accordance (and *bfr*-accordance, see Sect. 7) is not a precongruence with regard to open net composition $\oplus$. In addition to the models, we have $\Omega_X = \{[p_3]\}$ and $\Omega_Y = \{[p_7]\}$.

As in Sect. 4, we give a trace-based semantics for b-responsiveness. Then, we characterize the coarsest precongruence that is contained in the *br*-accordance relation.

6.1 A trace-based semantics for bounded responsiveness

Our trace-based semantics for b-responsiveness of an open net N essentially combines the *stop*-semantics of Definition 3.4 and the b-bounded semantics of Definition 5.2. It consists of the set of all $bound_b$-violators, the flooded language, and the flooded *stop*-semantics.

Definition 6.4 (b-bounded *stop*-semantics). The *b-bounded stop-semantics* of a labeled net N is defined by the sets of traces

- $bound_b(N)$,
- $L_b(N)$, and
- $stop_b(N) = stop(N) \cup bound_b(N)$.

The following lemma implies that, for the purpose of characterizing *br*-controllers in Proposition 6.6 below, we could also work with a variant of the

b-bounded *stop*-semantics that contains stop-traces instead of $stop_b$-traces. This observation makes the proof of Proposition 6.6 easy. Still, we use the $stop_b$-traces in this semantics because it gives a better sufficient condition for br-accordance in Theorem 6.7.

Lemma 6.5. *For composable open nets N_1 and N_2 with $L_b(N_1) \cap bound_b(N_2) = \emptyset$ and $bound_b(N_1) \cap L_b(N_2) = \emptyset$, we have*

$$stop(N_1) \cap stop(N_2) = \emptyset \text{ iff } stop_b(N_1) \cap stop_b(N_2) = \emptyset\,.$$

Proof. $\Leftarrow$: Follows immediately from Definition 6.4.

$\Rightarrow$: We can write $stop_b(N_1) \cap stop_b(N_2)$ as a union of four intersections

$$\begin{aligned} stop_b(N_1) \cap stop_b(N_2) = \quad & (stop(N_1) \cap bound_b(N_2)) \\ & \cup (bound_b(N_1) \cap stop(N_2)) \\ & \cup (bound_b(N_1) \cap bound_b(N_2)) \\ & \cup (stop(N_1) \cap stop(N_2)), \end{aligned}$$

all of which are empty:

- $stop(N_1) \cap bound_b(N_2) = \emptyset$, by $L_b(N_1) \cap bound_b(N_2) = \emptyset$ and $stop(N_1) \subseteq L(N_1) \subseteq L_b(N_1)$
- $bound_b(N_1) \cap stop(N_2) = \emptyset$, by $bound_b(N_1) \cap L_b(N_2) = \emptyset$ and $stop(N_2) \subseteq L(N_2) \subseteq L_b(N_2)$
- $bound_b(N_1) \cap bound_b(N_2) = \emptyset$, by $bound_b(N_1) \cap L_b(N_2) = \emptyset$ and $bound_b(N_2) \subseteq L_b(N_2)$
- $stop(N_1) \cap stop(N_2) = \emptyset$ by assumption. □

Some open net C is a br-controller if and only if it is a b-controller and an r-controller. Thus, Corollary 5.5 and Proposition 3.7 in combination with Lemma 6.5 give the following characterization of b-responsiveness.

Proposition 6.6 (b-responsiveness vs. b-bounded *stop*-semantics). *For composable open nets N_1 and N_2 such that $N_1 \oplus N_2$ is a closed net, we have*

$$N_1 \text{ and } N_2 \text{ are } b\text{-responsive} \quad \text{iff} \quad \begin{array}{l} stop_b(N_1) \cap stop_b(N_2) = \emptyset \text{ and} \\ L_b(N_1) \cap bound_b(N_2) = \emptyset \text{ and} \\ bound_b(N_1) \cap L_b(N_2) = \emptyset\,. \end{array}$$

Inclusion of the b-bounded *stop*-semantics defines a refinement relation which implies br-accordance.

Theorem 6.7 (b-bounded *stop*-inclusion implies br-accordance). *For two interface-equivalent open nets Impl and Spec, we have*

$$\begin{array}{l} bound_b(Impl) \subseteq bound_b(Spec) \text{ and} \\ L_b(Impl) \subseteq L_b(Spec) \text{ and} \\ stop_b(Impl) \subseteq stop_b(Spec) \end{array} \quad \text{implies} \quad Impl \sqsubseteq_{br,acc} Spec\,.$$

Proof. Proof by contraposition. Consider an open net C such that $\mathit{Impl} \oplus C$ and $\mathit{Spec} \oplus C$ are closed nets. Assume that C is not a br-controller of Impl. Then, Impl and C are not b-responsive by Definition 6.3, and we have $\mathit{stop}_b(\mathit{Impl}) \cap \mathit{stop}_b(C) \neq \emptyset$, $L_b(\mathit{Impl}) \cap \mathit{bound}_b(C) \neq \emptyset$, or $\mathit{bound}_b(\mathit{Impl}) \cap L_b(C) \neq \emptyset$ by Proposition 6.6. Because of the assumed inclusions, we have $\mathit{stop}_b(\mathit{Spec}) \cap \mathit{stop}_b(C) \neq \emptyset$, $L_b(\mathit{Spec}) \cap \mathit{bound}_b(C) \neq \emptyset$, or $\mathit{bound}_b(\mathit{Spec}) \cap L_b(C) \neq \emptyset$. Again with Proposition 6.6, we see that Spec and C are not b-responsive; that is, C is not a br-controller of Spec. □

The converse of Theorem 6.7 does not hold in general, as the next example shows.

Example 6.2. Recall that the open net D' br-accords with the open net D for every $b \in \mathbb{N}_+$ (see Example 6.1) although the language of D' is not contained in the language of D. For instance, we have $\mathit{sf} \in L_b(D') \setminus L_b(D)$. However, sf cannot be used reliably by any br-controller of D.

The cause of the counterexample in Example 6.2 and, thus, the reason why the converse of Theorem 6.7 does not hold is that br-accordance ignores those parts of open nets Impl and Spec that cannot be used reliably—that is, those markings that cannot be covered in the composition with any br-controller. In contrast, standard trace-based semantics compare the two open nets as a whole.

That standard language inclusion can be too strict has been observed for a stronger criterion than b-responsiveness in [21,5,23]. Mooij et al. [23] propose two solutions to overcome this problem. The first idea is to restrict the class of open nets considered to those where every place and transition can be covered. The second idea is to encode the covering nature of br-accordance in the trace-based semantics. In the following, we work out the latter idea in the present setting.

6.2 A coverable trace-based semantics for b-responsiveness

We aim to encode the covering nature of br-accordance in the b-bounded *stop*-semantics. To achieve this, we introduce a set that captures all *br-uncoverable* traces; that is, traces w that cannot be executed by (the environment of) any br-controller of N, regardless whether w can be executed in $\mathit{env}(N)$ or not.

Replacing in every trace set of the b-bounded *stop*-semantics of an open net N the set of bound_b-violators by the set of br-uncoverable traces yields the *coverable b-bounded stop-semantics* of N. This semantics differs from the previous trace semantics, as the br-uncoverable traces are an *external* characterization—they quantify over all br-controllers of N. The latter does not cause a problem, because we can calculate this set.

Definition 6.8 (coverable b-bounded *stop*-semantics). Let N be an open net. A word $w \in (I \uplus O)^*$ is a *br-uncoverable trace* of N if there does not exist a br-controller C of N with $w \in L_b(C)$. The *coverable b-bounded stop-semantics* of N is defined by the three sets of traces

- $\mathit{uncov}_{br}(N) = \{w \in (I \uplus O)^* \mid w \text{ is an } br\text{-uncoverable trace of } N\}$,

- $uL_{br}(N) = L(N) \cup uncov_{br}(N)$, and
- $ustop_{br}(N) = stop(N) \cup uncov_{br}(N)$.

Example 6.3. As mentioned in Example 6.1, for any bound b, there exists no br-controller of D or D' that marks the place s. Thus, $s \in uncov_{br}(D)$ and $s \in uncov_{br}(D')$, for instance.

By Proposition 6.6, a $bound_b$-violator of an open net is an br-uncoverable trace of N. So we directly conclude that the set of $bound_b$-violators of N is contained in the set of br-uncoverable traces of N. As a result, the coverable b-bounded *stop*-semantics extends the b-bounded *stop*-semantics by flooding more traces: $bound_b$-violators and br-uncoverable traces.

Lemma 6.9 (b-bounded *stop*-semantics is included). *For any open net N, we have*

- $bound_b(N) \subseteq uncov_{br}(N)$,
- $L_b(N) \subseteq uL_{br}(N)$, *and*
- $stop_b(N) \subseteq ustop_{br}(N)$.

Inclusion of the flooded stop-traces, the flooded language, and the br-uncoverable traces defines a refinement relation. We show that an open net *Impl* br-accords with an open net *Spec* if the respective traces of *Impl*'s coverable b-bounded *stop*-semantics are included in the respective traces of *Spec*'s coverable b-bounded *stop*-semantics. For the proof, we use the following two lemmata. The first lemma states that for every trace w, which is neither a trace nor a br-uncoverable trace of N, there exists a br-controller of N containing w in its set of $bound_b$-violators.

Lemma 6.10. *Let N be an open net. If $w \notin uL_{br}(N)$, then there exists a br-controller C of N with $w \in bound_b(C)$.*

Proof. Let $w \notin uL_{br}(N)$ with $w = w_1 \dots w_k$ for $j = 1, \dots, k$, $w_j \in I_N \uplus O_N$. As $w \notin uncov_{br}(N)$, there exists a br-controller C of N with $w \in L_b(C)$ by Definition 6.8. If $w \notin bound_b(C)$, we construct from w and C a br-controller $N_w^{bound} \oplus C'$ of N with $bound_b$-violator w as follows: In a first step, we construct an open net N_w that basically shifts all tokens from N to C, and vice versa. Moreover, N_w tracks whether a firing sequence in C is a prefix of w, and subsequently moving a token in N_w from an initially marked place p_0 to a place p_k. Intuitively, a token on a place p_j means we already encountered the trace $w_1 \dots w_j$. For shifting, we introduce several interface transitions in N_w for each interface place in N. In a second step, if and only if the place p_k is marked—that is, we encountered the trace $w_1 \dots w_k = w$—a "disaster" transition $t_{disaster}$ will be enabled, which may produce an unlimited number of tokens onto an inner place $p_{disaster}$. The latter construction yields the open net N_w^{bound}. This construction guarantees that w is a $bound_b$-violator of $N_w^{bound} \oplus C'$.

Let $I' = \{i' \mid i \in I_C\}$ and $O' = \{o' \mid o \in O_C\}$ be "fresh" copies of I_C and O_C. We derive the open net $C' = (P_C, T_C, F'_C, m_C, I', O', \emptyset)$ from C by renaming the interface of C and adjusting the flow relation accordingly. We define the open net $N_w = (P', T', F', m_{N_w}, O_N \uplus O_{C'}, I_N \uplus I_{C'}, \emptyset)$ with

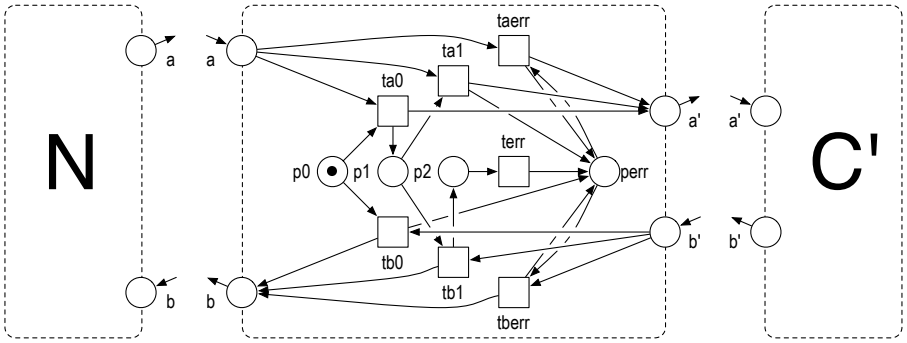


Fig. 11: Illustration of the construction of the open net N_w for N with $O_N = \{a\}$, $I_N = \{b\}$, and $w = ab$.

- $P' = \{p_i \mid 0 \leq i \leq k\} \uplus \{p_{err}\}$,
- $T' = \{t_i^x \mid 0 \leq i \leq k-1 \wedge x \in O_N \uplus I_N\} \uplus \{t_{err}^x \mid x \in O_N \uplus I_N\} \uplus \{t_{err}\}$,
- $F' = \{(x, t_i^x), (t_i^x, x'), (p_i, t_i^x) \mid 0 \leq i \leq k-1 \wedge x \in O_N\}$
  $\uplus \{(x', t_i^x), (t_i^x, x), (p_i, t_i^x) \mid 0 \leq i \leq k-1 \wedge x \in I_N\}$
  $\uplus \{(t_i^x, p_{i+1}) \mid 0 \leq i \leq k-1 \wedge x \in O_N \uplus I_N \wedge x = w_{i+1}\}$
  $\uplus \{(t_i^x, p_{err}) \mid 0 \leq i \leq k-1 \wedge x \in O_N \uplus I_N \wedge x \neq w_{i+1}\}$
  $\uplus \{(x, t_{err}^x), (t_{err}^x, x'), (p_{err}, t_{err}^x), (t_{err}^x, p_{err}) \mid x \in O_N\}$
  $\uplus \{(x', t_{err}^x), (t_{err}^x, x), (p_{err}, t_{err}^x), (t_{err}^x, p_{err}) \mid x \in I_N\}$
  $\uplus \{(p_k, t_{err}), (t_{err}, p_{err})\}$,
- $m_{N_w} = [p_0]$.

Figure 11 illustrates the construction of N_w. Clearly, we have $L(C) = L(N_w \oplus C')$, $bound_b(C) = bound_b(N_w \oplus C')$, and $stop(C) = stop(N_w \oplus C')$. Therefore, the open net $N_w \oplus C'$ is a br-controller of N by Proposition 6.6.

The places $p_0, \ldots, p_k, p_{err}$ together always carry one token, and p_k gets marked if and only if we encountered a trace whose prefix is w. Next, we extend N_w to an open net N_w^{bound} by adding a *disaster* transition $t_{disaster}$. The transition $t_{disaster}$ may produce an unlimited number of tokens onto an inner place $p_{disaster}$ of N_w^{bound}. Formally, we define the open net $N_w^{bound} = (P' \uplus \{p_{disaster}\}, T' \uplus \{t_{disaster}\}, F' \uplus F'', m_{N_w}, O_N \uplus O_{C'}, I_N \uplus I_{C'}, \emptyset)$ with $F'' = \{(p_k, t_{disaster}), (t_{disaster}, p_k), (t_{disaster}, p_{disaster})\}$. Thus, $w \in bound_b(N_w^{bound} \oplus C')$ for the br-controller $N_w^{bound} \oplus C'$ of N. □

The second lemma states that for every trace w of N, which is neither a stop-trace nor br-uncoverable, there exists a br-controller of N containing w in its set of stop-traces.

Lemma 6.11. *Let N be an open net. If $w \in L(N) \setminus \mathit{ustop}_{br}(N)$, then there exists a br-controller C of N with $w \in \mathit{stop}(C)$.*

Proof. Let $w \in L(N) \setminus \mathit{ustop}_{br}(N)$ with $w = w_1 \dots w_k$ for $j = 1, \dots, k$, $w_j \in I_N \uplus O_N$. As $w \notin \mathit{uncov}_{br}(N)$, there exists a br-controller C of N with $w \in L_b(C)$ by Definition 6.8. Then $w \in L(C) \setminus \mathit{bound}_b(C)$; otherwise, C is not a br-controller of N by Proposition 6.6. As w is no stop-trace of N, there exists an output $o \in O_N$ such that $m_{env(N)} \xRightarrow{wo}$. We conclude that $wo \in L(N)$ and $wo \in L(C)$, thus wo is not a bound_b-violator of N by Proposition 6.6.

Like in the proof of Lemma 6.10, we construct a br-controller $N_w \oplus C'$ of N with *stop*-trace w: We track whether a firing sequence in C is a prefix of w by composing C with another open net N_w, and subsequently moving a token in N_w from an initially marked place p_0 to a place p_k. Later, if and only if the place p_k is marked—that is, we encountered the trace $w_1 \dots w_k = w$—we prevent any output from N_w, but allow input to N_w. This will make w a *stop*-trace. Once N_w receives one input—for example, $o \in O_N$ as discussed above—we make N_w transparent again.

Let $I' = \{i' \mid i \in I_C\}$ and $O' = \{o' \mid o \in O_C\}$ be "fresh" copies of I_C and O_C. We derive the open net $C' = (P_C, T_C, F'_C, m_C, I', O', \emptyset)$ from C by renaming the interface of C and adjusting the flow relation accordingly. We define the open net $N_w = (P', T', F', m_{N_w}, O_N \uplus O_{C'}, I_N \uplus I_{C'}, \emptyset)$ with

- $P' = \quad \{p_i \mid 0 \le i \le k\}$
  $\quad \uplus \{p_{err}\}$,
- $T' = \quad \{t_i^x \mid 0 \le i \le k \wedge x \in O_N\}$
  $\quad \uplus \{t_i^x \mid 0 \le i \le k-1 \wedge x \in I_N\}$
  $\quad \uplus \{t_{err}^x \mid x \in O_N \uplus I_N\}$
  $\quad \uplus \{t_{err}\}$,
- $F' = \quad \{(x, t_i^x), (t_i^x, x'), (p_i, t_i^x) \mid 0 \le i \le k \wedge x \in O_N\}$
  $\quad \uplus \{(x', t_i^x), (t_i^x, x), (p_i, t_i^x) \mid 0 \le i \le k-1 \wedge x \in I_N\}$
  $\quad \uplus \{(t_i^x, p_{i+1}) \mid 0 \le i \le k-1 \wedge x \in O_N \uplus I_N \wedge x = w_{i+1}\}$
  $\quad \uplus \{(t_i^x, p_{err}) \mid 0 \le i \le k-1 \wedge x \in O_N \uplus I_N \wedge x \ne w_{i+1}\}$
  $\quad \uplus \{(t_k^x, p_{err}) \mid x \in O_N\}$
  $\quad \uplus \{(x, t_{err}^x), (t_{err}^x, x'), (p_{err}, t_{err}^x), (t_{err}^x, p_{err}) \mid x \in O_N\}$
  $\quad \uplus \{(x', t_{err}^x), (t_{err}^x, x), (p_{err}, t_{err}^x), (t_{err}^x, p_{err}) \mid x \in I_N\}$
  $\quad \uplus \{(p_k, t_{err}), (t_{err}, p_{err})\}$,
- $m_{N_w} = [p_0]$.

Figure 12 illustrates the construction of N_w. Clearly, we have $w \in \mathit{stop}(N_w \oplus C')$, and the open net $N_w \oplus C'$ is a br-controller of N. □

Finally, we show that br-accordance coincides with the refinement relation defined by inclusion of the coverable b-bounded *stop*-semantics.

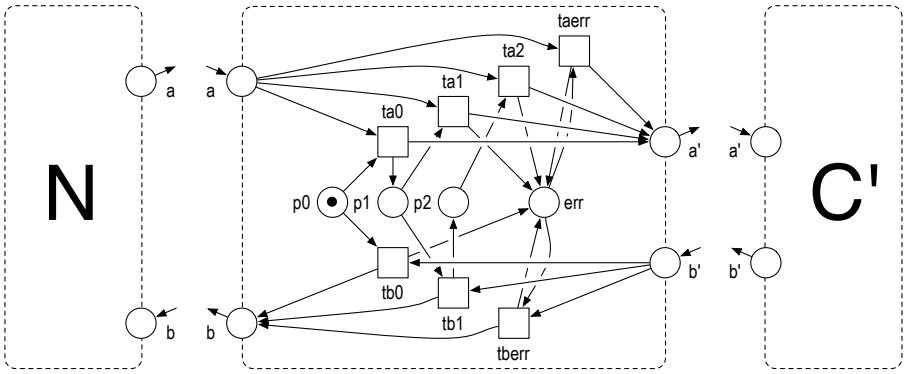


Fig. 12: Illustration of the construction of the open net N_w for N with $O_N = \{a\}$, $I_N = \{b\}$, and $w = ab$.

Theorem 6.12 (coverable b-bounded *stop*-inclusion vs. br-accordance). *For two interface-equivalent open nets Impl and Spec, we have*

$$Impl \sqsubseteq_{br,acc} Spec \quad \textit{iff} \quad \begin{array}{l} uncov_{br}(Impl) \subseteq uncov_{br}(Spec), \\ uL_{br}(Impl) \subseteq uL_{br}(Spec), \text{ and} \\ ustop_{br}(Impl) \subseteq ustop_{br}(Spec)\,. \end{array}$$

Proof. $\Rightarrow$: Let $w \notin uncov_{br}(Spec)$; that is, there exists a br-controller C of *Spec* with $w \in L_b(C)$. Clearly, C is a br-controller of *Impl* by $Impl \sqsubseteq_{br,acc} Spec$ and, thus, $w \notin uncov_{br}(Impl)$. This proves $uncov_{br}(Impl) \subseteq uncov_{br}(Spec)$.

Let $w \in uL_{br}(Impl) \setminus uncov_{br}(Impl)$ and assume $w \notin uL_{br}(Spec)$. There exists a br-controller C of *Spec* with $w \in bound_b(C)$ by Lemma 6.10. Clearly, C is not a br-controller of *Impl* by Proposition 6.6, and we have a contradiction to our assumption that $Impl \sqsubseteq_{br,acc} Spec$. Thus, $w \in uL_{br}(Spec)$.

Let $w \in ustop_{br}(Impl) \setminus uncov_{br}(Impl)$ and assume $w \notin ustop_{br}(Spec)$. Then, $w \in stop(Impl) \subseteq L(Impl)$ and $w \in L(Spec)$, as $uL_{br}(Impl) \subseteq uL_{br}(Spec)$ has been shown already. We can construct a br-controller C of *Spec* with $w \in stop(C)$ by Lemma 6.11. Clearly, C is not a br-controller of *Impl* by Proposition 6.6, and we have a contradiction to our assumption that $Impl \sqsubseteq_{br,acc} Spec$. Thus, $w \in ustop_{br}(Spec)$.

$\Leftarrow$: Proof by contraposition. Assume that the three inclusions hold and that C is not a br-controller of *Impl*. We show that C is not a br-controller of *Spec* either. If *Impl* and C are not b-responsive, we have $L_b(Impl) \cap bound_b(C) \neq \emptyset$, or $bound_b(Impl) \cap L_b(C) \neq \emptyset$, or $stop_b(Impl) \cap stop_b(C) \neq \emptyset$ by Proposition 6.6. We consider each case separately:

- $w \in L_b(Impl) \cap bound_b(C)$: Then $w \in L_b(Impl) \subseteq uL_{br}(Impl) \subseteq uL_{br}(Spec)$ by Lemma 6.9 and assumption. If $w \in L(Spec)$ then C is not a br-controller of *Spec* by $w \in bound_b(C)$ and Proposition 6.6; otherwise, if $w \in uncov_{br}(Spec)$, then C is not a br-controller of *Spec* by $w \in bound_b(C) \subseteq L_b(C)$ and Definition 4.3.

- $w \in \mathit{bound}_b(\mathit{Impl}) \cap L_b(C)$: Then $w \in \mathit{bound}_b(\mathit{Impl}) \subseteq \mathit{uncov}_{br}(\mathit{Impl}) \subseteq \mathit{uncov}_{br}(\mathit{Spec})$ by Lemma 6.9 and assumption. Then C is not a br-controller of *Spec* by Definition 4.3.
- $w \in \mathit{stop}_b(\mathit{Impl}) \cap \mathit{stop}_b(C)$: Then, $w \in \mathit{stop}_b(\mathit{Impl}) \subseteq \mathit{ustop}_{br}(\mathit{Impl}) \subseteq \mathit{ustop}_{br}(\mathit{Spec})$ by Lemma 6.9 and assumption. If $w \in \mathit{stop}(\mathit{Spec})$ then C is not a br-controller of *Spec* by $w \in \mathit{stop}_b(C)$ and Proposition 6.6; otherwise, if $w \in \mathit{uncov}_{br}(\mathit{Spec})$, then C is not a br-controller of *Spec* by $w \in \mathit{stop}_b(C) \subseteq L_b(C)$ and Definition 4.3. □

Example 6.4. Example 6.2 shows that $D' \sqsubseteq_{br,acc} D$ although $\mathit{sf} \in L_b(D') \setminus L_b(D)$. This difference is hidden in the coverable b-bounded *stop*-semantics due to flooding: $\mathit{sf} \in uL_{br}(D') \subseteq uL_{br}(D)$.

Despite the external characterization of the trace set uncov_{br}, we can compute the coverable b-bounded *stop*-semantics of an open net N by using the notion of a most permissive controller [37], which is a controller that can visit all the markings that can be visited using any controller. So, the coverable markings of an open net are the markings that can be visited by a most permissive controller. The construction is not the focus of this article and hence not shown.

As shown in Example 6.1, br-accordance is not a precongruence. The next section characterizes the coarsest precongruence that is contained in br-accordance.

6.3 Deriving the coarsest precongruence for b-responsiveness

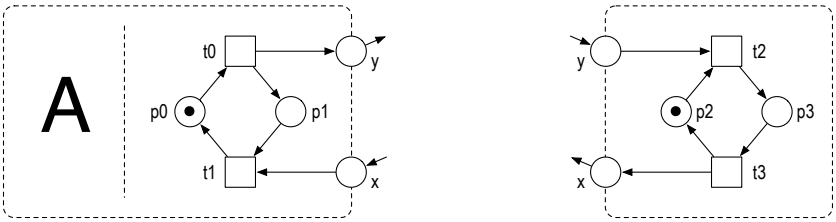
To derive the coarsest precongruence for b-responsiveness, we need to cope with the restriction to b-boundedness and, therefore, add information about bound_b-violators to the $\mathcal{F}^+$-semantics in Definition 3.9. For br-accordance, we observed that it does not imply L_b-inclusion (see Example 6.2, whereas it implies uL_{br}-inclusion (see Theorem 6.12). Therefore, one could wonder whether we should consider uncoverable traces rather than bound_b-violators. The following lemma shows that for the coarsest precongruence the situation is different because this precongruence implies L_b-inclusion.

Lemma 6.13 ($\sqsubseteq^c_{br,acc}$ implies $\sqsubseteq_{b,acc}$). *For interface-equivalent open nets Impl and Spec, we have*

$$\mathit{Impl} \sqsubseteq^c_{br,acc} \mathit{Spec} \text{ implies } \mathit{Impl} \sqsubseteq_{b,acc} \mathit{Spec}\,.$$

Proof. Assume $\mathit{Impl} \not\sqsubseteq_{b,acc} \mathit{Spec}$ and consider a b-controller A of *Spec* which is not a b-controller of *Impl*. Then the construction in Fig. 13 shows that open net C in Fig. 13b is a br-controller of $\mathit{Spec} \oplus A'$ but not of $\mathit{Impl} \oplus A'$. This contradicts that $\mathit{Impl} \sqsubseteq^c_{br,acc} \mathit{Spec}$. □

In the light of Lemma 6.13, we add information about bound_b-violators to the $\mathcal{F}^+$-semantics in Definition 3.9. The resulting b-bounded $\mathcal{F}^+$-semantics consists of the b-bounded semantics and the $\mathcal{F}^+$-semantics extended with all tree failures (w, X) where w is a trace of bound_b and X is any subset of the language.



(a) Open net A' (b) Open net C

Fig. 13: Construction for proof of Lemma 6.13

Definition 6.14 (b-bounded $\mathcal{F}^+$-semantics). For a labeled net N, we define $\mathit{fbound}_b(N) = \{(w, X) \mid w \in \mathit{bound}_b(N) \wedge X \in \mathcal{P}((I \uplus O)^+)\}$ and the *b-bounded $\mathcal{F}^+$-semantics* of N by

1. $\mathit{bound}_b(N)$ and
2. $\mathcal{F}_b^+(N) = \mathcal{F}^+(N) \cup \mathit{fbound}_b(N)$.

The $\mathcal{F}_b^+$-refinement relation combines b-accordance (see Definition 5.1) and the $\mathcal{F}^+$-refinement relation (see Definition 3.14). It is a pleasant surprise that this combination works.

Definition 6.15 ($\mathcal{F}_b^+$-refinement). For two interface-equivalent labeled nets *Impl* and *Spec*, *Impl $\mathcal{F}_b^+$-refines Spec*, denoted by $\mathit{Impl} \sqsubseteq_{\mathcal{F}_b^+} \mathit{Spec}$, if

1. $\mathit{bound}_b(\mathit{Impl}) \subseteq \mathit{bound}_b(\mathit{Spec})$ and
2. $\forall (w, X) \in \mathcal{F}_b^+(\mathit{Impl}) : \exists x \in \{\varepsilon\} \cup \downarrow X : (wx, x^{-1}X) \in \mathcal{F}_b^+(\mathit{Spec})$.

We say (w, X) is *dominated* by $(wx, x^{-1}X)$. For two interface-equivalent open nets *Impl* and *Spec*, we define $\mathit{Impl} \sqsubseteq_{\mathcal{F}_b^+} \mathit{Spec}$, if $\mathit{env}(\mathit{Impl}) \sqsubseteq_{\mathcal{F}_b^+} \mathit{env}(\mathit{Spec})$.

Example 6.5. Consider again the open nets D and D'. For any bound b, we have $(sf, \emptyset) \in \mathcal{F}_b^+(D')$ but $(sf, \emptyset) \notin \mathcal{F}_b^+(D)$; thus, D' does not $\mathcal{F}_b^+$-refine D.

The next lemma shows that the b-bounded $\mathcal{F}^+$-semantics refines the b-bounded semantics as it should in view of Lemma 6.13.

Lemma 6.16 ($\mathcal{F}_b^+$-refinement implies language inclusion). *For labeled nets Impl and Spec, we have*

$$\mathit{Impl} \sqsubseteq_{\mathcal{F}_b^+} \mathit{Spec} \text{ implies } L_b(\mathit{Impl}) \subseteq L_b(\mathit{Spec}).$$

Proof. Let $w \in L_b(\mathit{Impl})$. Then $(w, \emptyset) \in \mathcal{F}_b^+(\mathit{Impl})$ by Definition 6.14, and $(w, \emptyset) \in \mathcal{F}_b^+(\mathit{Spec})$ by Definition 6.15 which immediately implies $w \in L_b(\mathit{Spec})$ by Definition 6.14. □

If we consider the composition of two open nets N_1 and N_2, then its b-bounded $\mathcal{F}^+$-semantics coincides with that of the parallel composition of the two environments, $env(N_1) \Uparrow env(N_2)$.

Lemma 6.17. *For composable open nets N_1 and N_2, we have*

$$\mathcal{F}_b^+(N_1 \oplus N_2) = \mathcal{F}_b^+(env(N_1) \Uparrow env(N_2))\,.$$

Proof. Follows directly from Lemma 2.12: If one net has a $bound_b$-violator w due to marking m, then the other net can reach an agreeing marking m' with trace w; thus w is also a $bound_b$-violator for the other net (where, if the latter is $env(N_1) \Uparrow env(N_2)$, we fire 'interface transitions' to make the markings agree strongly if necessary). Likewise, if one net has a tree failure (w, X) due to marking m, then the other net can reach an agreeing marking m' with trace w. If w is a $bound_b$-violator in the other net, then X is a refusal set by definition of the b-bounded $\mathcal{F}^+$-semantics. Otherwise, if some trace $v \in X$ could be performed from m', this would also be possible from m due to bisimilarity, yielding a contradiction. Thus, (w, X) is also a tree failure of the other net. □

We want to show that $\mathcal{F}_b^+$-refinement is a precongruence on open nets for composition operator $\oplus$, and for this we will use the precongruence results for $\mathcal{F}^+$-refinement and b-accordance. First, we characterize the b-bounded $\mathcal{F}^+$-semantics for labeled net composition and hiding, and then combine these results for open net composition.

Lemma 6.18 (b-bounded $\mathcal{F}^+$-semantics for labeled net composition). *For composable labeled nets N_1 and N_2, we have*

$$\begin{aligned}\mathcal{F}_b^+(N_1 \| N_2) = \quad & \{(w, X) \mid \exists (w_1, X_1) \in \mathcal{F}_b^+(N_1), (w_2, X_2) \in \mathcal{F}_b^+(N_2) : \\ & \qquad w \in w_1 \| w_2 \wedge \forall x \in X : \\ & \qquad\qquad x \in x_1 \| x_2 \textit{ implies } x_1 \in X_1 \vee x_2 \in X_2\} \\ & \cup \mathit{fbound}_b(N_1 \| N_2)\,.\end{aligned}$$

Proof. We write E for $N_1 \| N_2$.

$\subseteq$: Let $(w, X) \in \mathcal{F}_b^+(E)$. If w is not a $bound_b$-violator of E, then $(w, X) \in \mathcal{F}^+(E)$ by Definition 6.14, and we conclude with Proposition 3.11 and Definition 6.14 that it is contained in the first set on the right hand side. If w is a $bound_b$-violator of E, then $(w, X) \in \mathit{fbound}_b(E)$ by Definition 6.14.

$\supseteq$: Let $i = 1, 2$. If both $(w_i, X_i) \in \mathcal{F}^+(N_i)$, then $(w, X) \in \mathcal{F}^+(E)$ by Proposition 3.11 and $\mathcal{F}^+(E) \subseteq \mathcal{F}_b^+(E)$. Assume now that at least one tree failure (w_i, X_i) is not contained in the respective $\mathcal{F}^+$-semantics. Then trace w_i is a $bound_b$-violator by Definition 6.14 and so is w by Proposition 5.3(1), because $w_{3-i} \in L_b(N_{3-i})$ as argued in the proof of Lemma 6.16. Thus, $(w, X) \in \mathit{fbound}_b(E) \subseteq \mathcal{F}_b^+(E)$ due to Definition 6.14. Furthermore, $\mathit{fbound}_b(E) \subseteq \mathcal{F}_b^+(E)$ by Definition 6.14. □

In the next lemma, we consider a labeled net N/A, $A \subseteq \Sigma$ and we use $\phi(w)$ to denote $w|_{\Sigma \setminus A}$. We canonically extend the notion of $\phi(w)$ pointwise to sets of traces.

Lemma 6.19. *For any labeled net N and any label set $A \subseteq \Sigma_N^*$, we have*

$$\mathcal{F}_b^+(N/A) = \{(\phi(w), X) \mid (w, \phi^{-1}(X)) \in \mathcal{F}_b^+(N)\}\,.$$

Proof. Follows by Proposition 5.3(5) and Proposition 3.12. □

Recall Proposition 5.3(7), which shows how to determine $\mathit{bound}_b(N_1 \oplus N_2)$ from the bound_b- and the L_b-semantics of the components; in our setting, we can read off the L_b-semantics from the b-bounded $\mathcal{F}^+$-semantics; see the proof of Lemma 6.16. The next proposition similarly characterizes the b-bounded $\mathcal{F}^+$-semantics for open net composition, also using the b-bounded semantics. In the proofs of the following two results, we will denote the first set on the right-hand side in Lemma 6.18 by $\mathcal{F}1_b^+(N_1, N_2)$.

Proposition 6.20 (b-bounded $\mathcal{F}^+$-semantics for open net composition). *For composable open nets N_1 and N_2, we have*

$$\begin{aligned}\mathcal{F}_b^+(N_1 \oplus N_2) = \quad & \{(w, X) \mid \exists (w_1, X_1) \in \mathcal{F}_b^+(N_1), (w_2, X_2) \in \mathcal{F}_b^+(N_2) : \\ & \qquad w \in w_1 \Uparrow w_2 \wedge \forall x \in X : \\ & \qquad\qquad x \in x_1 \Uparrow x_2 \textit{ implies } x_1 \in X_1 \vee x_2 \in X_2\} \\ & \cup \mathit{fbound}_b(N_1 \oplus N_2)\,.\end{aligned}$$

Proof. According to Lemma 6.17, we can consider $\mathcal{F}_b^+(\mathit{env}(N_1) \Uparrow \mathit{env}(N_2))$ instead of $\mathcal{F}_b^+(N_1 \oplus N_2)$. Because $\Uparrow$ is $\|$ followed by hiding, we can determine this set by applying hiding (according to Proposition 5.3(5) and Lemma 6.19) to the right-hand side of Lemma 6.18. As a result, $\mathcal{F}1_b^+(N_1, N_2)$ turns into the first set in the present proposition, just as Proposition 3.13 results from Proposition 3.11 in combination with Proposition 3.12. More easily, $\mathit{fbound}_b(N_1 \| N_2)$ is analogously translated into $\mathit{fbound}_b(N_1 \oplus N_2)$ according to Proposition 5.3(1) and 5.3(7). □

Next, we show that $\mathcal{F}_b^+$-refinement is a precongruence on labeled nets for the composition operator $\|$. We will use the following three saturation conditions, which also hold for the $\mathcal{F}^+$-semantics [29]:

SAT1: $(w, X) \in \mathcal{F}_b^+(N), X' \subseteq X$ implies $(w, X') \in \mathcal{F}_b^+(N)$
SAT2: $(w, X) \in \mathcal{F}_b^+(N) \wedge \forall z \in Z : (wz, z^{-1}X) \notin \mathcal{F}_b^+(N)$ implies $(w, X \cup Z) \in \mathcal{F}_b^+(N)$
SAT3: $(w, X) \in \mathcal{F}_b^+(N)$ implies $(w, \uparrow X) \in \mathcal{F}_b^+(N)$

In [29], $\|$ is defined directly on sets of tree failures, taking essentially the equation in Proposition 3.11 as definition. Then, just from the saturation conditions, it is shown that $\mathcal{F}^+$-refinement (defined as in Definition 6.15(2)) is a precongruence for $\|$. We will make use of this, although this defining equation does not match Lemma 6.18, but just gives $\mathcal{F}1_b^+(N_1, N_2)$.

Lemma 6.21. *$\mathcal{F}_b^+$-refinement is a precongruence on labeled nets for the composition operator $\|$.*

Proof. To see the three saturation conditions, consider first some $(w, X) \in \mathcal{F}^+(N) \subseteq \mathcal{F}_b^+(N)$. Then, SAT1 and SAT3 follow directly from [29], as does SAT2 once we observe that $(wz, z^{-1}X) \notin \mathcal{F}_b^+(N)$ implies $(wz, z^{-1}X) \notin \mathcal{F}^+(N)$. Second, consider a tree failure (w, X) with $w \in \mathit{bound}_b(N)$; here, all three conditions are immediate because $(w, Y) \in \mathcal{F}_b^+(N)$ for any $Y \in \mathcal{P}((I \uplus O)^+)$.

Now let $\mathit{Impl} \sqsubseteq_{\mathcal{F}_b^+} \mathit{Spec}$ and C be a composable labeled net for *Impl* and *Spec*. We have to check the two items of Definition 6.15 in order to prove that $\mathit{Impl} \| C \sqsubseteq_{\mathcal{F}_b^+} \mathit{Spec} \| C$.

The first item follows from Proposition 5.3(1) because our assumption implies $\mathit{bound}_b(\mathit{Impl}) \subseteq \mathit{bound}_b(\mathit{Spec})$ and—due to Lemma 6.16—$L_b(\mathit{Impl}) \subseteq L_b(\mathit{Spec})$.

For the second item, we first consider some $(w, X) \in \mathcal{F}1_b^+(\mathit{Impl}, C)$. We observe that, due to Definition 6.15(2), $\mathcal{F}_b^+(\mathit{Impl})$ is related to $\mathcal{F}_b^+(\mathit{Spec})$ in the sense of $\mathcal{F}^+$-refinement. Thus, according to the precongruence result of [29], $\exists x \in \{\varepsilon\} \cup \downarrow X : (wx, x^{-1}X) \in \mathcal{F}1_b^+(\mathit{Spec}, C) \subseteq \mathcal{F}_b^+(\mathit{Spec} \| C)$. Second, we consider some $(w, X) \in \mathit{fbound}_b(\mathit{Impl} \| C)$. This time, due to $\mathit{bound}_b(\mathit{Impl} \| C) \subseteq \mathit{bound}_b(\mathit{Spec} \| C)$, we even have $(w, X) \in \mathit{fbound}_b(\mathit{Spec} \| C) \subseteq \mathcal{F}_b^+(\mathit{Spec} \| C)$—that is, Definition 6.15(2) is satisfied taking $x = \varepsilon$. □

We show that $\mathcal{F}_b^+$-refinement for labeled nets is preserved under hiding.

Lemma 6.22. *$\mathcal{F}_b^+$-refinement is a precongruence on labeled nets for hiding.*

Proof. Let *Impl* and *Spec* be labeled nets with $\mathit{Impl} \sqsubseteq_{\mathcal{F}_b^+} \mathit{Spec}$, and $A \subseteq \Sigma^*$. Then Proposition 5.3(5) directly implies Definition 6.15(1) for Impl/A and Spec/A. Furthermore, the characterization in Lemma 6.19 corresponds to the defining equation for hiding in [29], so Definition 6.15(2) is inherited from the precongruence result in [29] for $\mathcal{F}^+$-refinement and hiding. □

Now we directly get the first main result of this section that also the $\mathcal{F}_b^+$-refinement relation is a precongruence for composition operator $\oplus$.

Theorem 6.23 (precongruence). *$\mathcal{F}_b^+$-refinement is a precongruence for the composition operator $\oplus$.*

Proof. This is now completely analogous to the proof of Theorem 3.15, except that here the semantics associates two sets with a net and we use Lemma 6.21 and 6.22 on the basis of Lemma 6.17 and Lemma 5.4. □

With the next theorem, we prove that the coarsest precongruence which is contained in the *br*-accordance and $\mathcal{F}_b^+$-refinement coincide.

Theorem 6.24 (precongruence and $\mathcal{F}_b^+$-refinement coincide). *For two interface-equivalent open nets Impl and Spec, we have*

$$\mathit{Impl} \sqsubseteq^c_{br,acc} \mathit{Spec} \quad \textit{iff} \quad \mathit{Impl} \sqsubseteq_{\mathcal{F}_b^+} \mathit{Spec}\,.$$

Proof. $\Rightarrow$: By Lemma 6.13, $\sqsubseteq^c_{br,acc}$ implies $\sqsubseteq_{b,acc}$ from which we conclude $bound_b$-inclusion. It remains to show $\mathcal{F}^+_b$-inclusion. Let $(w, X) \in \mathcal{F}^+_b(Impl)$. If $w \in bound_b(Impl) \subseteq bound_b(Spec)$ (by $bound_b$-inclusion), then $(w, X) \in \mathcal{F}^+_b(Spec)$ and we are done. Otherwise, we use net N in Fig. 6 and the open net C as in the proof of (the reverse implication of) Theorem 3.17. Following the argumentation in the proof of Theorem 3.17, we have that C is not a br-controller of $Impl \oplus N$ so it is not a br-controller of $Spec \oplus N$. We distinguish three cases: If $w \in bound_b(Spec)$, then $(w, X) \in \mathcal{F}^+_b(Spec)$ by Definition 6.14. If $wu \in bound_b(Spec)$ with $u \in \downarrow X$, then $(wu, u^{-1}X) \in \mathcal{F}^+_b(Spec)$ by Definition 6.14. Otherwise, we use the argumentation in the proof of Theorem 3.17 to conclude that for some u, $(wu, u^{-1}X) \in \mathcal{F}^+(Spec) \subseteq \mathcal{F}^+_b(Spec)$.

$\Leftarrow$: We show that $Impl \sqsubseteq_{\mathcal{F}^+_b} Spec$ implies $bound_b$-inclusion, L_b-inclusion and $stop_b$-inclusion from which we conclude by Theorem 6.7 that $Impl \sqsubseteq_{br,acc} Spec$. The latter, in turn, also shows that $Impl \sqsubseteq_{\mathcal{F}^+_b} Spec$ implies $Impl \sqsubseteq^c_{br,acc} Spec$ with Theorem 6.23 and the definition of $\sqsubseteq^c_{br,acc}$.

$bound_b$-inclusion follows directly from Definition 6.15, and L_b-inclusion follows from Lemma 6.16. It remains to show $stop_b$-inclusion: Let O be the set of output places of $Impl$ and, equivalently, of $Spec$. We have $w \in stop_b(Impl)$ iff $(w, O) \in \mathcal{F}^+_b(Impl)$ by Definition 6.14. Then $Impl \sqsubseteq_{\mathcal{F}^+_b} Spec$ implies $(w, O) \in \mathcal{F}^+_b(Spec)$, thus $w \in stop_b(Spec)$. □

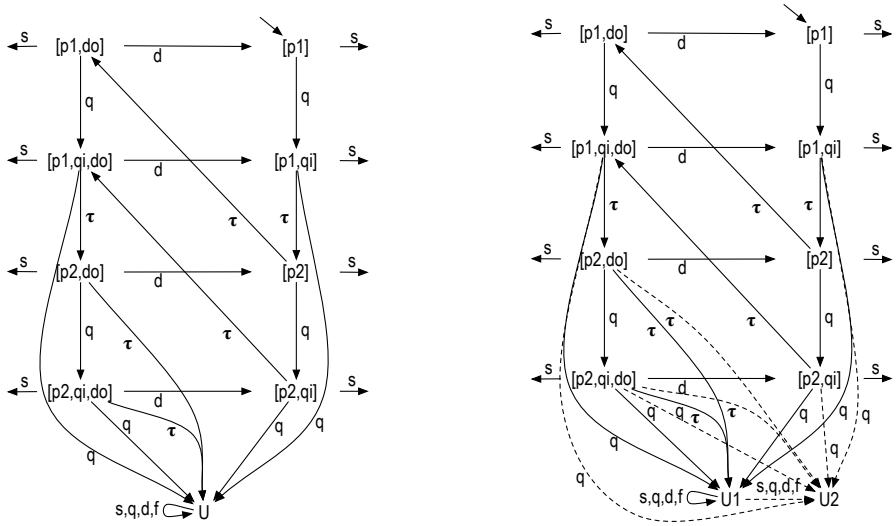
Note that $stop$-inclusion does not hold because $(w, O) \in \mathcal{F}^+_b(Spec)$ does not imply $w \in stop(Spec)$.

Example 6.6. For our example open nets D and D', we have shown in Example 6.1 that D' br-accords with D, but D' does not $\mathcal{F}^+_b$-refine D as illustrated in Example 6.5. As a consequence, $D' \sqsubseteq^c_{br,acc} D$ does not hold.

6.4 Decidability of $\mathcal{F}^+_b$-refinement

In this section, we show that checking $\mathcal{F}^+_b$-refinement is decidable. For this, we work with the reachability graph $RG(N)$ of a labeled net N and the set $B(N) \subseteq M_N$ of bound violations of N. Actually, our approach works for arbitrary labeled transition systems and the difference between inputs and outputs does not matter. To stress the generality of our approach and to abstract from the not so relevant details of open nets, we assume we are given an arbitrary LTS S (as used in [29, Theorem 61]) where the state set of S is partitioned into $Q \uplus B(S)$ with a finite set Q of states that are reachable from the initial state m_S (i.e., $M_N - B(S)$ in N) and a possibly infinite set $B(S)$ of *bad* states (i.e., $B(N)$ in N). For such an LTS S, we can define $bound_b(S)$, $\mathcal{F}^+(S)$ and $\mathcal{F}^+$-refinement (as for labeled nets). Let $B\mathcal{F}^+_b(S) = \{(w, X) \mid w \in bound_b(S), X \subseteq (I \cup O)^+\}$ and $\mathcal{F}^+_b(S) = \mathcal{F}^+(S) \cup B\mathcal{F}^+_b(S)$; define $\mathcal{F}^+_b$-refinement as in Definition 6.15.

Checking $\mathcal{F}^+_b$-refinement entails checking both items of Definition 6.15. The first item of Definition 6.15—that is, checking $bound_b$-inclusion—is decidable because we can represent the languages $bound_b(S)$ as a finite automaton. To



(a) Automaton for $bound_1(D)$ (b) Automaton $U_{12}(D)$

Fig. 14: Sketches of the two finite automata used in the proof of Theorem 6.28. A transition involving s is indicated by an arrow without sink.

this end, we merge all bad states into a single state U; more formally, we add a new state U with a self-loop for each action $a \in I \uplus O$, and replace each arc $m \xrightarrow{a} m'$ by $m \xrightarrow{a} U$ whenever $m \notin B(S)$ while $m' \in B(S)$; finally, we restrict the LTS to the states reachable from m_S (which are all in Q) and consider only U as final state. Now checking $bound_b$-inclusion reduces to checking language inclusion for two finite automata.

Example 6.7. Figure 14a sketches the automaton that represents $bound_1(D)$. Parts of the automaton in Fig. 14a that can only be reached with an s-transition are not shown. For example, we have $\{qq, qqqq, qqf\} \subseteq bound_1(D)$. The marking $[p_1, q^i, q^i]$ is not 1-bounded; therefore, the transition $[p_1, q^i] \xrightarrow{q} [p_1, q^i, q^i]$ in $RG(env(D))$ was replaced with $[p_1, q^i] \xrightarrow{q} U$. The traces qq and $qqqq$ are strict $bound_1$-violators, as they lead from the initial state $[p_1]$ to the final state U while visiting state U only once (for trace $qqqq$ with three τ-transitions in-between). The traces $qqqq$ and qqf visit U more than once.

To decide refinement of the tree failures—that is, the second item of Definition 6.15—we shall use the construction of Rensink and Vogler [29, Theorem 61] for deciding $\mathcal{F}^+$-refinement for finite-state systems. As checking $\mathcal{F}^+$-refinement is known to be decidable [29, Theorem 61], we can conclude that checking $\mathcal{F}_b^+$-refinement is decidable, too.

The automaton we constructed above for S also represents the language $L_b(S)$ if we regard all states as final. But it is not suitable for representing

$\mathcal{F}_b^+(S)$, since we cannot distinguish the $bound_b$-violators that can be performed from those that have only been added as continuations; thus, the refusal sets are not represented properly. Therefore, we propose a different finite-state representation of $\mathcal{F}_b^+(S)$ on which checking $\mathcal{F}^+$-refinement coincides with checking $\mathcal{F}_b^+$-refinement for S.

We construct the finite LTS $U_{12}(S)$ as follows. The state set of $U_{12}(S)$ is $Q \uplus \{U_1, U_2\}$, the start state is m_S, $B(U_{12}(S)) = \{U_1, U_2\}$ is the set of bound violations, and the transition relation is defined as

$$\{m \xrightarrow{a} m' \mid m, m' \in Q, a \in I \cup O \cup \{\tau\}, m \xrightarrow{a} m' \text{ in } S\}$$
$$\cup \{m \xrightarrow{a} U_1, m \xrightarrow{a} U_2 \mid m \in Q, \exists m' \in B(S), a \in I \cup O \cup \{\tau\} : m \xrightarrow{a} m' \text{ in } S\}$$
$$\cup \{U_1 \xrightarrow{a} U_1, U_1 \xrightarrow{a} U_2 \mid a \in I \cup O\}.$$

Example 6.8. Figure 14b shows a part of the automaton $U_{12}(D)$. For example, we have $(qqqq, \{f\}) \in \mathcal{F}_b^+(D) \cap \mathcal{F}^+(U_{12}(D))$, because the trace $qqqq$ may lead to the state U_2 and then refuse f. Further, we have $(q, \{qqf\}) \in \mathcal{F}_b^+(D) \setminus \mathcal{F}^+(U_{12}(D))$—observe f can never fire in $env(D)$, while any state reached by q in $U_{12}(D)$ can reach U_1 with qq and then add f. There is $q \in {\downarrow}\{qqf\} \setminus \{qqf\}$ and the trace qq is a $bound_1$-violator of D.

The next lemma, gives three observations about $U_{12}(S)$.

Lemma 6.25. *Let S be an LTS. Then the following facts hold for $U_{12}(S)$.*

1. *$w \in bound_b(S)$ iff $m_N \stackrel{w}{\Longrightarrow} U_1$ iff $m_N \stackrel{w}{\Longrightarrow} U_2$.*
2. *$B\mathcal{F}_b^+(S) \subseteq \mathcal{F}^+(U_{12}(S))$ (i.e., U_2 can refuse all $X \subseteq (I \cup O)^+$)*
3. *$\mathcal{F}^+(U_{12}(S)) \subseteq \mathcal{F}_b^+(S)$ and $(w, X) \in \mathcal{F}_b^+(S) \setminus \mathcal{F}^+(U_{12}(S))$ implies $\exists u \in {\downarrow} X \setminus X$ such that $wu \in bound_b(S)$*

Proof. The first item follows immediately from the definition of $U_{12}(S)$, and the second item is an implication of the first item. Consider the third item. The sets agree on those (w, X) with $w \in bound_b(S)$ by Item 2. So consider $w \notin bound_b(S)$. If $(w, X) \in \mathcal{F}^+(U_{12}(S))$ due to $m_S \stackrel{w}{\Longrightarrow} m$, then we also have $m_S \stackrel{w}{\Longrightarrow} m$ in S with the same run. In $U_{12}(S)$, m could only have more traces due to runs using U_1, so it can only refuse less. Thus, $(w, X) \in \mathcal{F}^+(S)$ and inclusion follows.

If $(w, X) \in \mathcal{F}_b^+(S)$ due to m, but $(w, X) \notin \mathcal{F}^+(U_{12}(S))$, then this must be due to a run from m that touches U_1. Assume this happens for the first time after $u \neq \varepsilon$; that is, we have $m_N \stackrel{w}{\Longrightarrow} m \stackrel{u}{\Longrightarrow} U_1 \stackrel{u'}{\Longrightarrow}$ with $uu' \in X$. Thus, $wu \in bound_b(S)$ and $u \in {\downarrow} X$. As $m_N \stackrel{w}{\Longrightarrow} m \stackrel{u}{\Longrightarrow}$ also in S, we further have $u \notin X$. □

The following lemma gives another prerequisite for the decidability proof.

Lemma 6.26. *For $X, Y \in \mathcal{P}(\Sigma^*)$, let $x \in {\downarrow} Y \cup \{\varepsilon\}$ and $X = x^{-1}Y$. Then*

1. *$u \in {\downarrow} X$ implies $xu \in {\downarrow} Y$*
2. *$u \notin X$ implies $xu \notin Y$*

Proof. (1) Due to $uu' \in X$ and $xuu' \in Y$.

(2) Suppose $xu \in Y$. Then $u \in X$ and we have a contradiction. □

With the next lemma and the subsequent theorem, we return to speaking about open nets – though we could as well talk about two LTS. The lemma shows that deciding $\mathcal{F}_b^+$-refinement for two open nets *Impl* and *Spec* reduces to checking $\mathcal{F}^+$-refinement of their automata $U_{12}(Spec)$ and $U_{12}(Impl)$.

Lemma 6.27. *For two open nets Impl and Spec such that* $bound_b(Impl) \subseteq bound_b(Spec)$*, we have*

$$Impl \sqsubseteq_{\mathcal{F}_b^+} Spec \text{ iff } U_{12}(Impl) \sqsubseteq_{\mathcal{F}^+} U_{12}(Spec)\,.$$

Proof. $\Rightarrow$: Each $(v, Y) \in \mathcal{F}^+(U_{12}(Impl)) \subseteq \mathcal{F}_b^+(Impl)$ (by Lemma 6.25(3)) is dominated by some $(w, X) \in \mathcal{F}_b^+(Spec)$ due to some $x \in \downarrow Y \cup \{\varepsilon\}$, i.e., $X = x^{-1}Y$, $w = vx$. If $(w, X) \in \mathcal{F}^+(U_{12}(Impl))$, we are done. So assume otherwise and consider $u \in \downarrow X \setminus X$ according to Lemma 6.25(3). Then $xu \in \downarrow Y \setminus Y$ by Lemma 6.26 and $(vxu, (xu)^{-1}Y) \in \mathcal{F}^+(U_{12}(Spec))$ because $vxu \in bound_b(Spec)$ and $\varepsilon \notin (xu)^{-1}Y$. Hence, (v, Y) is also covered in this case.

$\Leftarrow$: Each $(w, X) \in \mathcal{F}^+(U_{12}(Impl))$ is dominated by $\mathcal{F}^+(U_{12}(Spec)) \subseteq \mathcal{F}_b^+(Spec)$ by Lemma 6.25(3). So consider $(w, X) \in \mathcal{F}_b^+(Impl) \setminus \mathcal{F}^+(U_{12}(Impl))$ and respectively $u \in \downarrow X \setminus X$ according to Lemma 6.25(3). Then, $(wu, u^{-1}X) \in \mathcal{F}_b^+(Spec)$ because $wu \in bound_b(Impl) \subseteq bound_b(Spec)$ and $\varepsilon \notin u^{-1}X$. Thus, (w, X) is also dominated in this case. □

With Lemma 6.27, we have shown:

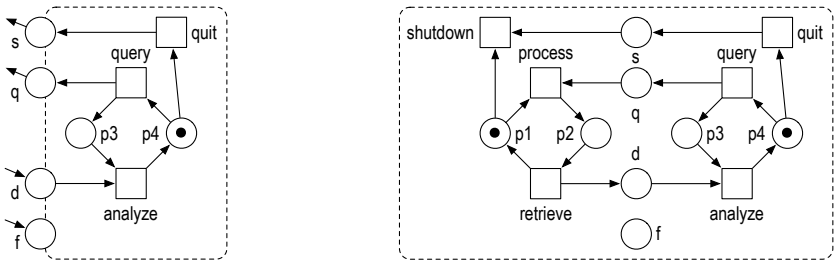
Theorem 6.28 ($\mathcal{F}_b^+$-refinement is decidable). *For two interface-equivalent open nets Impl and Spec, checking whether* $Impl \sqsubseteq_{\mathcal{F}_b^+} Spec$ *is decidable.*

7 Bounded nets and final markings

In this section, we consider open nets with final markings and whose composition is bounded. We refer to this notion of responsiveness as *bf*-responsiveness.

Definition 7.1 (*bf*-responsiveness). Let N_1 and N_2 be composable open nets. A marking of $N_1 \oplus N_2$ is *bf-responsive* if it is *f*-responsive and *b*-bounded. Open nets N_1 and N_2 are *bf*-responsive if their composition $N_1 \oplus N_2$ is a closed net and every reachable marking in $N_1 \oplus N_2$ is *bf-responsive*.

Two open nets are *bf*-responsive if and only if they are *f*-responsive and their composition is *b*-bounded. In the presence of final markings, we can prove a stronger semantical characterization of responsiveness than in Proposition 6.2. Due to *bf*-responsiveness each net has always the chance to send a message or the composition terminates.



(a) Open net U' (b) Open net $D \oplus U'$

Fig. 15: Open net U' modeling another user of open net D in Fig. 8a and their composition $D \oplus U'$. In addition to the models, we have $\Omega_{U'} = \Omega_{D \oplus U'} = \{[]\}$.

Proposition 7.2. *Let open nets N_1 and N_2 be bf-responsive. Then, from any reachable marking m, markings m_1 and m_2 are reachable such that $m_1 \xrightarrow{t_1}$ with $t_1^\bullet \cap O_1 \neq \emptyset$ and $m_2 \xrightarrow{t_2}$ with $t_2^\bullet \cap O_2 \neq \emptyset$, or a final marking is reachable.*

Proof. Direct consequence of Proposition 6.2 and Definitions 7.1 and 4.1. □

Again, we redefine the notion of a controller and of accordance for this variant of responsiveness to *bfr*-controller and *bfr*-accordance. Also *bfr*-accordance shall turn out not to be a precongruence; thus, we also introduce its coarsest precongruence.

Definition 7.3 (*bfr*-controller, *bfr*-accordance). An open net C is a *bfr-controller* of an open net N if N and C are *bf*-responsive.

For interface-equivalent open nets *Impl* and *Spec*, *Impl bfr-accords with Spec*, denoted by $Impl \sqsubseteq_{bfr,acc} Spec$, if for all open nets C holds: C is a *bfr*-controller of *Spec* implies C is a *bfr*-controller of *Impl*.

We denote the coarsest precongruence contained in $\sqsubseteq_{bfr,acc}$ by $\sqsubseteq^c_{bfr,acc}$.

Example 7.1. The open net U' in Fig. 15a represents another user of the database server D in Fig. 8a. It has the same functionality as the open net U in Fig. 8b, but may additionally decide to quit and shut down the database (output place s). The open nets D and U' are composable; their composition $D \oplus U'$ is a closed net, which is depicted in Fig. 15b. U' is not a *br*-controller of D because the nonresponsive marking $[\,]$ is reachable in $D \oplus U'$, but U' is a *bfr*-controller of D because $[\,]$ is a final marking of $D \oplus U'$. Moreover, U' is not a *br*-controller of D' because $D' \oplus U'$ can reach the non-*bf*-responsive marking $[f]$ (see Proposition 7.2). Thus, although D' *br*-accords with D (see Example 6.1) it does not *bfr*-accord with D.

As shown in Example 6.1, D *br*-accords with D'. No *bfr*-controller of D' can send s as otherwise this can lead to the marking $[f]$. Thus, we also have D *bfr*-accords with D'.

Like the previous responsiveness variants, bounded final accordance is not a precongruence either. D *bfr*-accords with D', but $D \oplus Y$ does not *bfr*-accord with $D' \oplus Y$, because open net X is a *bfr*-controller of $D' \oplus Y$ but not of $D \oplus Y$ (see Fig. 10).

We continue by giving a trace-based semantics for *bf*-responsiveness. Then, we formalize the coarsest precongruence, which is contained in the *bfr*-accordance relation.

7.1 A trace-based semantics for bounded final responsiveness

Our trace-based semantics for *bf*-responsiveness of an open net N combines the *stop dead*-semantics of Definition 4.1 and the b-bounded *stop*-semantics of Definition 6.1. The resulting *b-bounded stop dead-semantics* consists of four sets of traces: $bound_b$-violators, the language, stop-traces, and dead-traces.

Definition 7.4 (b-bounded *stop dead*-semantics). The *b-bounded stop dead-semantics* of a labeled net N is defined by the sets of traces

- $bound_b(N)$,
- $L_b(N)$,
- $stop_b(N)$, and
- $dead_b(N) = dead(N) \cup bound_b(N)$.

For the purpose of characterizing *bfr*-controllers, we could also work with a variant of the b-bounded *stop dead*-semantics that contains stop-traces and dead-traces instead of the flooded version. This is similar to the characterization of *br*-controllers in Sect. 6.1. Still, we use $stop_b$-traces and $dead_b$-traces in this semantics because it gives a better sufficient condition for *bfr*-accordance in Theorem 7.7 below.

Lemma 7.5. *For composable open nets N_1 and N_2 with $L_b(N_1) \cap bound_b(N_2) = \emptyset$ and $bound_b(N_1) \cap L_b(N_2) = \emptyset$, we have*

$$\begin{array}{lll} stop(N_1) \cap dead(N_2) = \emptyset \text{ and} & \text{iff} & stop_b(N_1) \cap dead_b(N_2) = \emptyset \text{ and} \\ dead(N_1) \cap stop(N_2) = \emptyset & & dead_b(N_1) \cap stop_b(N_2) = \emptyset\,. \end{array}$$

Proof. $\Leftarrow$: Follows immediately from Definition 7.4.

$\Rightarrow$: We can write $stop_b(N_1) \cap dead_b(N_2)$ as a union of four intersections

$$\begin{aligned} stop_b(N_1) \cap dead_b(N_2) = \quad & (stop(N_1) \cap dead(N_2)) \\ & \cup\, (bound_b(N_1) \cap dead(N_2)) \\ & \cup\, (stop(N_1) \cap bound_b(N_2)) \\ & \cup\, (bound_b(N_1) \cap bound_b(N_2)), \end{aligned}$$

all of which are empty:

- $stop(N_1) \cap dead(N_2) = \emptyset$ by assumption

- $bound_b(N_1) \cap dead(N_2) = \emptyset$, by $bound_b(N_1) \cap L_b(N_2) = \emptyset$ (Corollary 5.5) and $dead(N_2) \subseteq L(N_2) \subseteq L_b(N_2)$
- $stop(N_1) \cap bound_b(N_2) = \emptyset$, by $L_b(N_1) \cap bound_b(N_2) = \emptyset$ (Corollary 5.5) and $stop(N_1) \subseteq L(N_1) \subseteq L_b(N_1)$
- $bound_b(N_1) \cap bound_b(N_2) = \emptyset$, by $bound_b(N_1) \cap L_b(N_2) = \emptyset$ (Corollary 5.5) and $bound_b(N_2) \subseteq L_b(N_2)$

A similar argument applies to $dead_b(N_1) \cap stop_b(N_2) = \emptyset$. □

An open net C is a *bfr*-controller of an open net N if and only if it is an *fr*-controller and a *b*-controller of N. Thus, by Proposition 4.6 and Corollary 5.5 in combination with Lemma 7.5, we give the following characterization of *bf*-responsiveness.

Proposition 7.6 (*bf*-responsiveness vs. *b*-bounded *stopdead*-semantics). *Let N_1 and N_2 be composable open nets such that $N_1 \oplus N_2$ is a closed net. Then*

$$N_1 \text{ and } N_2 \text{ are } bf\text{-responsive} \quad \text{iff} \quad \begin{array}{l} stop_b(N_1) \cap dead_b(N_2) = \emptyset \text{ and} \\ dead_b(N_1) \cap stop_b(N_2) = \emptyset \text{ and} \\ L_b(N_1) \cap bound_b(N_2) = \emptyset \text{ and} \\ bound_b(N_1) \cap L_b(N_2) = \emptyset\,. \end{array}$$

Inclusion of the *b*-bounded *stopdead*-semantics defines a refinement relation which implies *bf*-responsiveness.

Theorem 7.7 (*b*-bounded *stopdead*-inclusion implies *bfr*-accordance). *For two interface-equivalent open nets Impl and Spec, we have*

$$\begin{array}{l} bound_b(Impl) \subseteq bound_b(Spec) \text{ and} \\ L_b(Impl) \subseteq L_b(Spec) \text{ and} \\ stop_b(Impl) \subseteq stop_b(Spec) \text{ and} \\ dead_b(Impl) \subseteq dead_b(Spec) \end{array} \quad \text{implies} \quad Impl \sqsubseteq_{bfr,acc} Spec\,.$$

Proof. Proof by contraposition. Consider an open net C such that $Impl \oplus C$ and, equivalently, $Spec \oplus C$ are closed nets. Otherwise, C is neither a *bfr*-controller of *Impl* nor of *Spec*. Assume that C is not a *bfr*-controller of *Impl*. Then, *Impl* and C are not *bf*-responsive by Definition 7.3, and we have either $stop_b(Impl) \cap dead_b(C) \neq \emptyset$, $dead_b(Impl) \cap stop_b(C) \neq \emptyset$, $L_b(Impl) \cap bound_b(C) \neq \emptyset$, or $bound_b(Impl) \cap L_b(C) \neq \emptyset$ by Proposition 7.6. Because of the assumed inclusions, we have either $stop_b(Spec) \cap dead_b(C) \neq \emptyset$, $dead_b(Spec) \cap stop_b(C) \neq \emptyset$, $L_b(Spec) \cap bound_b(C) \neq \emptyset$, or $bound_b(Spec) \cap L_b(C) \neq \emptyset$. Again with Proposition 7.6, we see that *Spec* and C are not *bf*-responsive; that is, C is not a *bfr*-controller of *Spec* and thus $Impl \sqsubseteq_{bfr,acc} Spec$. □

As for Theorem 6.7, the converse of Theorem 7.7 does not hold in general either, because bounded final responsiveness considers only the reliable part of open nets.

Example 7.2. Consider the open nets D and D', this time, with the empty sets of final markings. No *bfr*-controller of D will send an s because this would eventually lead to firing of *shutdown*, yielding a non-*bf*-responsive marking where neither p_1 nor p_2 contains a token. Thus, we conclude that D' *bfr*-accords with D. Now we can apply the same trace sf as in Example 6.2 and obtain $sf \in L_b(D') \setminus L_b(D)$.

For this reason, we adapt the coverable b-bounded *stop*-semantics from the previous section.

7.2 A coverable trace-based semantics for bounded final responsiveness

We combine the idea of the coverable b-bounded *stop*-semantics in Definition 6.8 and the *stopdead*-semantics in Definition 4.3 yielding the *coverable b-bounded stopdead-semantics*. For this, we define the notion of an *bfr*-uncoverable trace. As for the language and the stop-traces in the coverable b-bounded *stop*-semantics, we also flood the dead-traces in our new semantics.

Definition 7.8 (coverable b-bounded *stopdead*-semantics). Let N be an open net. A word $w \in (I \uplus O)^*$ is a *bfr-uncoverable trace* of N if there does not exist a *bfr*-controller C of N with $w \in L_b(C)$. The *coverable b-bounded stopdead-semantics* of N is defined by the sets of traces

- $uncov_{bfr}(N) = \{w \in (I \uplus O)^* \mid w \text{ is a } bfr\text{-uncoverable trace of } N\}$,
- $uL_{bfr}(N) = L(N) \cup uncov_{bfr}(N)$,
- $ustop_{bfr}(N) = stop(N) \cup uncov_{bfr}(N)$, and
- $udead_{bfr}(N) = dead(N) \cup uncov_{br}(N)$.

Example 7.3. We showed in Example 7.1 that D *bfr*-accords with D'. Now observe that $s \in stop_b(D)$ but $s \notin stop_b(D')$; that is, $stop_b$-inclusion fails. However, no *bfr*-controller of D' has trace s in its language; otherwise, D' may reach the non-*bfr*-responsive marking $[\,]$. Thus, we have $s \in uncov_{bfr}(D')$ and s is also in the flooded stop-set of D', i.e., $s \in ustop_{br}(D')$.

By Proposition 7.6, a $bound_b$-violator of an open net is an *bfr*-uncoverable trace of N. So we directly conclude that the set of $bound_b$-violators of N is contained in the set of *bfr*-uncoverable traces of N. As a result, the coverable b-bounded *stopdead*-semantics includes the b-bounded *stopdead*-semantics.

Lemma 7.9 (b-bounded *stopdead*-semantics is included). *For any open net N, we have*

- $bound_b(N) \subseteq uncov_{bfr}(N)$,
- $L_b(N) \subseteq uL_{bfr}(N)$,
- $stop_b(N) \subseteq ustop_{bfr}(N)$, *and*
- $dead_b(N) \subseteq udead_{bfr}(N)$.

Next, we characterize *bfr*-accordance. For the proof, we restate Lemmata 6.10 and 6.11 for *bf*-responsiveness. In addition, we show that for every trace w of N, which is neither a dead-trace nor *bfr*-uncoverable, there exists a *bfr*-controller of N containing w in its set of stop-traces.

Lemma 7.10. *Let N be an open net. Then*

1. *If $w \notin uL_{bfr}(N)$, then there exists a bfr-controller C of N with $w \in bound_b(C)$.*
2. *If $w \in L(N) \setminus ustop_{bfr}(N)$, then there exists a bfr-controller C of N with $w \in dead(C)$.*
3. *If $w \in L(N) \setminus udead_{bfr}(N)$, then there exists a bfr-controller C of N with $w \in stop(C)$.*

Proof. (1) This is analogous to the proof of Lemma 6.10, except that C is a *bfr*-controller of N and we have to consider the final markings. The final markings of N_w^{bound} are $\Omega_{N_w^{bound}} = \{[p] \mid p \in \{p_0, \ldots, p_k, p_{err}\}\}$. That way we do not add additional dead-traces to $N_w^{bound} \oplus C'$ beside continuations of $w \in bound_b(N_w^{bound} \oplus C')$. Clearly, $N_w^{bound} \oplus C'$ is still a *bfr*-controller of N because of $w \notin uL_{br}(N)$ and Proposition 7.6.

(2) This is analogous to the proof of Lemma 6.11, except that C is a *bfr*-controller of N and we put $\Omega_{N_w} = \{[p] \mid p \in \{p_0, \ldots, p_{k-1}, p_{err}\}\}$. As $[p_k]$ is not a final marking of N_w, the stop-trace w of $N_w \oplus C'$ is a dead-trace of $N_w \oplus C'$.

(3) This is analogous to the proof of Lemma 6.11, except that C is a *bfr*-controller of N and we have to change N_w slightly: The introduced stop-trace w of $N_w \oplus C'$ may be a stop-trace of N, i.e., $w \in stop(N) \setminus dead(N)$. Thus, after N_w recognizes w, we buffer all messages from C' to N_w inside N_w and define all markings of N_w and C' as final markings. That way, we do not introduce an additional dead-trace of $N_w \oplus C'$. □

Theorem 7.11 (coverable b-bounded *stop dead*-inclusion vs. *bfr*-accordance). *For two interface-equivalent open nets Impl and Spec, we have*

$$Impl \sqsubseteq_{bfr,acc} Spec \quad \text{iff} \quad \begin{array}{l} uncov_{bfr}(Impl) \subseteq uncov_{bfr}(Spec), \\ uL_{bfr}(Impl) \subseteq uL_{bfr}(Spec), \\ ustop_{bfr}(Impl) \subseteq ustop_{bfr}(Spec), \text{ and} \\ udead_{bfr}(Impl) \subseteq udead_{bfr}(Spec)\,. \end{array}$$

Proof. $\Rightarrow$: For the first three trace inclusions, we apply the same argumentation as in the proof of Theorem 6.12 by replacing Proposition 6.6, Lemma 6.10, and Lemma 6.11 with Proposition 7.6, Lemma 7.10(1) and(2), respectively.

Let $w \in udead_{bfr}(Impl) \setminus uncov_{bfr}(Impl)$ and assume $w \notin udead_{bfr}(Spec)$. Then, $w \in dead(Impl) \subseteq L(Impl)$ and $w \in L(Spec)$, as $uL_{bfr}(Impl) \subseteq uL_{bfr}(Spec)$ has been shown already. We can construct a *bfr*-controller C of *Spec* with $w \in stop(C)$ by Lemma 7.10(3). Clearly, C is not a *bfr*-controller of *Impl* by Proposition 7.6, and we have a contradiction to our assumption that $Impl \sqsubseteq_{bfr,acc} Spec$. Thus, $w \in udead_{bfr}(Spec)$.

$\Leftarrow$: Proof by contraposition. Assume that the four inclusions hold and that C is not a *bfr*-controller of *Impl*. We show that C is not a *bfr*-controller of *Spec* either.

Impl and C are not *bf*-responsive by Definition 7.3, and we have $stop_b(Impl) \cap dead_b(C) \neq \emptyset$, $dead_b(Impl) \cap stop_b(C) \neq \emptyset$, $L_b(Impl) \cap bound_b(C) \neq \emptyset$, or $bound_b(Impl) \cap L_b(C) \neq \emptyset$ by Proposition 7.6. We consider each case separately:

- $w \in stop_b(Impl) \cap dead_b(C)$: Then, $w \in ustop_{bfr}(Impl) \subseteq ustop_{bfr}(Spec)$ by Lemma 7.9 and assumption. If $w \in stop(Spec)$ then C is not a *bfr*-controller of *Spec* by Proposition 7.6; otherwise, if $w \in uncov_{bfr}(Spec)$, then C is not a *bfr*-controller of *Spec* by $w \in dead_b(C) \subseteq L_b(C)$ and Definition 7.8.
- $w \in dead_b(Impl) \cap stop_b(C)$: Then, $w \in udead_{bfr}(Impl) \subseteq udead_{bfr}(Spec)$ by Lemma 7.9 and assumption. If $w \in dead(Spec)$ then C is not a *bfr*-controller of *Spec* by Proposition 7.6; otherwise, if $w \in uncov_{bfr}(Spec)$, then C is not a *bfr*-controller of *Spec* by $w \in stop_b(C) \subseteq L_b(C)$ and Definition 7.8.
- $w \in L_b(Impl) \cap bound_b(C)$ or $w \in bound_b(Impl) \cap L_b(C)$: same argumentation as in the proof of Theorem 6.12 by replacing Proposition 6.6 and Lemma 6.9 with Proposition 7.6 and Lemma 7.9, respectively. □

Like the trace set $uncov_{br}$, also the trace set $uncov_{bfr}$ can be calculated using the notion of a most permissive controller. However, as this calculation is not the focus of this article, we do not show it.

7.3 Deriving the coarsest precongruence for bounded final responsiveness

In this section, we shall derive the coarsest precongruence for *bf*-responsiveness. To cope with the combination of f-responsiveness and b-boundedness, we add information about $bound_b$-violators to the $\mathcal{F}^+_{fin}$-semantics (Definition 4.8), similarly as we did to derive the b-bounded $\mathcal{F}^+$-semantics from the $\mathcal{F}^+$-semantics. The resulting semantics is the *b-bounded $\mathcal{F}^+_{fin}$-semantics*, where an element (w, X, Y) is called *b-bounded fintree failure*.

Definition 7.12 (b-bounded $\mathcal{F}^+_{fin}$-semantics). For a labeled net N, we define $finbound_b(N) = bound_b(N) \times \mathcal{P}(\Sigma^+) \times \mathcal{P}(\Sigma^*)$ and the *b-bounded $\mathcal{F}^+_{fin}$-semantics* of N by

- $bound_b(N)$ and
- $\mathcal{F}^+_{b,fin}(N) = \mathcal{F}^+_{fin}(N) \cup finbound_b(N)$.

The $\mathcal{F}^+_{b,fin}$-refinement relation combines the $\mathcal{F}^+_{fin}$-refinement relation (see Definition 4.9) and b-accordance (see Definition 5.1).

Definition 7.13 ($\mathcal{F}^+_{b,fin}$-refinement). For two interface-equivalent labeled nets *Impl* and *Spec*, *Impl $\mathcal{F}^+_{b,fin}$-refines Spec*, denoted by $Impl \sqsubseteq_{\mathcal{F}^+_{b,fin}} Spec$, if

1. $bound_b(Impl) \subseteq bound_b(Spec)$ and

2. $\forall (w, X, Y) \in \mathcal{F}^+_{b,fin}(Impl) : \exists x \in \{\varepsilon\} \cup \downarrow X \cup \downarrow Y : (wx, x^{-1}X, x^{-1}Y) \in \mathcal{F}^+_{b,fin}(Spec)$.

We say (w, X, Y) is *dominated* by $(wx, x^{-1}X, x^{-1}Y)$. For two interface-equivalent open nets *Impl* and *Spec*, we define $Impl \sqsubseteq_{\mathcal{F}^+_{b,fin}} Spec$, if $env(Impl) \sqsubseteq_{\mathcal{F}^+_{b,fin}} env(Spec)$.

If two open nets are in the $\mathcal{F}^+_{b,fin}$-refinement relation, then this implies inclusion of their bounded languages, stop-traces, and dead-traces.

Lemma 7.14. *For two labeled nets Impl and Spec, we have*

1. $Impl \sqsubseteq_{\mathcal{F}^+_{b,fin}} Spec$ *implies* $L_b(Impl) \subseteq L_b(Spec)$.
2. $Impl \sqsubseteq_{\mathcal{F}^+_{b,fin}} Spec$ *implies* $stop_b(Impl) \subseteq stop_b(Spec)$.
3. $Impl \sqsubseteq_{\mathcal{F}^+_{b,fin}} Spec$ *implies* $dead_b(Impl) \subseteq dead_b(Spec)$.

Proof. (1) Let $w \in L_b(Impl)$. Then $(w, \emptyset, \emptyset) \in \mathcal{F}^+_{b,fin}(Impl)$ by Definition 7.12 and $(w, \emptyset, \emptyset) \in \mathcal{F}^+_{b,fin}(Spec)$ by Definition 7.13, which immediately implies $w \in L_b(Spec)$ by Definition 7.12.

(2) Let $w \in stop_b(Impl)$. Then we use the proof of (the implication of) Theorem 4.17 by replacing *stop* by $stop_b$ to conclude that $w \in stop_b(Spec)$.

(3) Similar argumentation as for (2). □

Example 7.4. In Example 7.2, we showed for the open nets D and D' with the empty set of final markings that L_b-inclusion does not hold. Thus, with Lemma 7.14(1) we conclude that D' does not $\mathcal{F}^+_{b,fin}$-refine D.

Again, the composition of two open nets N_1 and N_2 has the same b-bounded $\mathcal{F}^+_{fin}$-semantics as the parallel composition of the two environments, $env(N_1) \Uparrow env(N_2)$.

Lemma 7.15. *For composable open nets N_1 and N_2, we have*

$$\mathcal{F}^+_{b,fin}(N_1 \oplus N_2) = \mathcal{F}^+_{b,fin}(env(N_1) \Uparrow env(N_2)) .$$

Proof. Follows directly from Lemma 2.12: If one net has a $bound_b$-violator w due to marking m, then the other net can reach an agreeing marking m' with trace w; thus w is also a $bound_b$-violator for the other net. Likewise, by applying the same argumentation as in the proof of Lemma 4.10, we conclude that if one net has a b-bounded fintree failure (w, X, Y) so does the other net. □

We shall show that $\mathcal{F}^+_{b,fin}$-refinement is a precongruence on open nets for $\oplus$, thereby using the precongruence results for $\mathcal{F}^+_{fin}$-refinement and b-accordance. We characterize the b-bounded $\mathcal{F}^+_{fin}$-semantics for labeled net composition and hiding and finally combine these results to determine the b-bounded $\mathcal{F}^+_{fin}$-semantics for open net composition. First, we consider the b-bounded $\mathcal{F}^+_{fin}$-semantics for the composition of two labeled nets. Here and below, we use $\pi_1(w)$ and $\pi_2(w)$ to denote $w|_{\Sigma_1}$ and $w|_{\Sigma_2}$ for labeled nets N_1 and N_2 with alphabets Σ_1 and Σ_2.

Lemma 7.16 ($\mathcal{F}^+_{b,fin}$-semantics for labeled net composition). *For two composable labeled nets N_1 and N_2, we have*

$$\begin{aligned}\mathcal{F}^+_{b,fin}(N_1\|N_2) = \quad & \{(w, X_1 \cup X_2, Y_1 \cup Y_2) \mid (\pi_1(w), \pi_1(X_1), \pi_1(Y_1)) \in \mathcal{F}^+_{b,fin}(N_1), \\ & \qquad\qquad (\pi_2(w), \pi_2(X_2), \pi_2(Y_2)) \in \mathcal{F}^+_{b,fin}(N_2)\} \\ & \cup \mathit{finbound}_b(N_1\|N_2)\,.\end{aligned}$$

Proof. We write E for $N_1\|N_2$.

$\subseteq$: Let $(w, X, Y) \in \mathcal{F}^+_{b,fin}(E)$. If w is not a bound_b-violator, then $(w, X, Y) \in \mathcal{F}^+_{fin}(E)$ by Definition 7.12, and we conclude with Lemma 4.11 and Definition 7.12 that it is contained in the first set on the right hand side. If w is a bound_b-violator of E, then $(w, X, Y) \in \mathit{finbound}_b(N_1\|N_2)$ by Definition 7.12.

$\supseteq$: Let $i = 1, 2$. If both $(\pi_i(w), \pi_i(X_i), \pi_i(Y_i)) \in \mathcal{F}^+_{fin}(N_i)$, then $(w, X_1 \cup X_2, Y_1 \cup Y_2) \in \mathcal{F}^+_{fin}(E)$ by Lemma 4.11 and $\mathcal{F}^+_{fin}(E) \subseteq \mathcal{F}^+_{b,fin}(E)$. Assume now that at least one b-bounded fintree failure $(\pi_i(w), \pi_i(X_i), \pi_i(Y_i))$ is not contained in the respective $\mathcal{F}^+_{fin}$-semantics. Then trace $\pi_i(w)$ is a bound_b-violator by Definition 7.12 and so is w by Proposition 5.3(1), because $\pi_{3-i}(w) \in L_b(N_{3-i})$ as argued in the proof of Lemma 7.14(1). Thus, $(w, X_1 \cup X_2, Y_1 \cup Y_2) \in \mathit{finbound}_b(E) \subseteq \mathcal{F}^+_{b,fin}(E)$ due to Definition 7.12. □

In the next lemma, we consider a labeled net N/A, $A \subseteq \Sigma$ and we use $\phi(w)$ to denote $w|_{\Sigma\setminus A}$. We canonically extend the notion of $\phi(w)$ pointwise to sets of traces.

Lemma 7.17. *For any labeled net N and any label set $A \subseteq \Sigma^*_N$, we have*

$$\mathcal{F}^+_{b,fin}(N/A) = \{(\phi(w), X, Y) \mid (w, \phi^{-1}(X), \phi^{-1}(Y)) \in \mathcal{F}^+_{b,fin}(N)\}\,.$$

Proof. Follows from Lemma 4.12. □

The next proposition characterizes the b-bounded $\mathcal{F}^+_{fin}$-semantics for open net composition. In the proofs of the following two results, we will denote the first set on the right-hand side in Lemma 7.16 by $\mathcal{F}1^+_{b,fin}(N_1, N_2)$.

Proposition 7.18 ($\mathcal{F}^+_{b,fin}$-semantics for open net composition). *For two composable open nets N_1 and N_2, we have*

$$\begin{aligned}\mathcal{F}^+_{b,fin}(N_1 \oplus N_2) = \quad & \{(w, X, Y) \mid \exists (w_1, X_1, Y_1) \in \mathcal{F}^+_{b,fin}(N_1), (w_2, X_2, Y_2) \in \mathcal{F}^+_{b,fin}(N_2) : \\ & \qquad w \in w_1 \Uparrow w_2 \wedge \forall x \in X, y \in Y : \\ & \qquad\qquad (x \in x_1 \Uparrow x_2 \textit{ implies } x_1 \in X_1 \vee x_2 \in X_2) \\ & \qquad\qquad \wedge\, (y \in y_1 \Uparrow y_2 \textit{ implies } y_1 \in Y_1 \vee y_2 \in Y_2)\} \\ & \cup \mathit{finbound}_b(N_1 \oplus N_2)\,.\end{aligned}$$

Proof. According to Lemma 7.15, we can consider $\mathcal{F}^+_{b,fin}(\mathit{env}(N_1) \Uparrow \mathit{env}(N_2))$ instead of $\mathcal{F}^+_{b,fin}(N_1 \oplus N_2)$. Because $\Uparrow$ is $\|$ followed by hiding, we can determine this set by applying hiding (according to Lemma 7.17) to the right-hand side of Lemma 7.16. As a result, $\mathcal{F}1^+_b(N_1, N_2)$ turns into the first set in the present proposition, just as Proposition 4.13 results from Lemma 4.11 with Lemma 4.12. More easily, $\mathit{finbound}_b(N_1\|N_2)$ is analogously translated into $\mathit{finbound}_b(N_1 \oplus N_2)$ according to Proposition 5.3(1) and 5.3(7). □

Lemma 7.19. *$\mathcal{F}_b^+$-refinement is a precongruence on labeled nets for composition operator $\|$.*

Proof. For this proof, we will use that the following four saturation conditions, which hold for the $\mathcal{F}_{fin}^+$-semantics (see Lemma 4.14), also hold for the b-bounded $\mathcal{F}_{fin}^+$-semantics:

SAT1: $(w,X,Y) \in \mathcal{F}_{b,fin}^+(N), X' \subseteq X, Y' \subseteq Y$ implies $(w,X',Y') \in \mathcal{F}_{b,fin}^+(N)$
SAT2: $(w,X,Y) \in \mathcal{F}_{b,fin}^+(N) \wedge \forall z \in Z : (wz, z^{-1}X, z^{-1}Y) \notin \mathcal{F}_{b,fin}^+(N)$ implies $(w, X \cup Z, Y \cup Z) \in \mathcal{F}_{b,fin}^+(N)$
SAT3: $(w,X,Y) \in \mathcal{F}_{b,fin}^+(N)$ implies $(w, \uparrow X, Y) \in \mathcal{F}_{b,fin}^+(N)$
SAT4: $(w,X,Y) \in \mathcal{F}_{b,fin}^+(N)$ implies $(w, X, X \cup Y) \in \mathcal{F}_{b,fin}^+(N)$

To see these conditions, consider first some $(w,X,Y) \in \mathcal{F}_{fin}^+(N) \subseteq \mathcal{F}_{b,fin}^+(N)$. Then, SAT1 – SAT4 follow directly from Lemma 4.14. Now, consider a b-bounded fintree failure (w,X,Y) with $w \in \mathit{bound}_b(N)$; here, all four conditions are immediate because $(w,X',Y') \in \mathcal{F}_{b,fin}^+(N)$ for any $X' \in \mathcal{P}((I \uplus O)^+)$ and $Y' \in \mathcal{P}((I \uplus O)^*)$.

The precongruence result for $\mathcal{F}_{fin}^+$-refinement holds for general sets of fintree failures (see Remark 4.2 below Lemma 4.14). We make use of this, although this defining equation does not match Lemma 7.16, but just gives $\mathcal{F}1_{b,fin}^+(N_1,N_2)$.

Now let $\mathit{Impl} \sqsubseteq_{\mathcal{F}_{b,fin}^+} \mathit{Spec}$ and C be a composable labeled net for Impl and Spec. We have to check the two items of Definition 6.15 in order to prove that $\mathit{Impl}\|C \sqsubseteq_{\mathcal{F}_{b,fin}^+} \mathit{Spec}\|C$.

The first item follows from Proposition 5.3(1) (which holds for labeled nets in general) because our assumption implies $\mathit{bound}_b(\mathit{Impl}) \subseteq \mathit{bound}_b(\mathit{Spec})$ as well as—due to Lemma 7.14(1)—$L_b(\mathit{Impl}) \subseteq L_b(\mathit{Spec})$.

For the second item, we first consider some $(w,X,Y) \in \mathcal{F}1_{b,fin}^+(\mathit{Impl}, C)$. We observe that, due to Definition 7.13(2), $\mathcal{F}_{b,fin}^+(\mathit{Impl})$ is related to $\mathcal{F}_{b,fin}^+(\mathit{Spec})$ in the sense of $\mathcal{F}_{fin}^+$-refinement. Thus, according to Lemma 4.14, $\exists x \in \{\varepsilon\} \cup \downarrow X \cup \downarrow Y : (wx, x^{-1}X, x^{-1}Y) \in \mathcal{F}1_{b,fin}^+(\mathit{Spec}, C) \subseteq \mathcal{F}_{b,fin}^+(\mathit{Spec}\|C)$. Second, we consider some $(w,X,Y) \in \mathit{finbound}_b(\mathit{Impl}\|C)$. This time, due to $\mathit{bound}_b(\mathit{Impl}\|C) \subseteq \mathit{bound}_b(\mathit{Spec}\|C)$, we even have $(w,X,Y) \in \mathit{finbound}_b(\mathit{Spec}\|C) \subseteq \mathcal{F}_{b,fin}^+(\mathit{Spec}\|C)$; that is, Definition 7.13(2) is satisfied taking $x = \varepsilon$. □

Lemma 7.20. *$\mathcal{F}_{b,fin}^+$-refinement is a precongruence on labeled nets for hiding.*

Proof. Let Impl and Spec be labeled nets with $\mathit{Impl} \sqsubseteq_{\mathcal{F}_{b,fin}^+} \mathit{Spec}$, and $A \subseteq \Sigma^*$. Then Lemma 7.17(1) directly implies Definition 7.13(1) for Impl/A and Spec/A. Furthermore, the characterization in Lemma 7.17(2) corresponds to the defining equation for hiding in Lemma 4.15, so Definition 7.13(2) is inherited from the precongruence result in Lemma 4.15 for $\mathcal{F}_{fin}^+$-refinement and hiding. □

Now we directly get the first main result of this section.

Theorem 7.21 (precongruence). *$\mathcal{F}^+_{b,fin}$-refinement is a precongruence on open nets for composition operator $\oplus$.*

Proof. This is now completely analogous to the proof of Theorem 4.16, except that here the semantics associates two sets with a net and we use Lemma 7.19 and 7.20 on the basis of Lemma 4.14 and Lemma 4.15. □

As for $\sqsubseteq^c_{br,acc}$ in Lemma 6.13, we show that $\sqsubseteq^c_{bfr,acc}$ implies $\sqsubseteq_{b,acc}$.

Lemma 7.22 ($\sqsubseteq^c_{bfr,acc}$ implies $\sqsubseteq_{b,acc}$). *For two open nets Impl and Spec, we have*

$$\mathit{Impl} \sqsubseteq^c_{bfr,acc} \mathit{Spec} \text{ implies } \mathit{Impl} \sqsubseteq_{b,acc} \mathit{Spec}\,.$$

Proof. Use the proof of Lemma 6.13 while assuming the set of final markings to be the empty set for all open nets. □

Next, we prove our second main result of this section: $\mathcal{F}^+_{b,fin}$-refinement coincides with the coarsest precongruence that is contained in the *bfr*-accordance relation.

Theorem 7.23 (precongruence and $\mathcal{F}^+_{b,fin}$-refinement coincide). *For two interface-equivalent open nets Impl and Spec, we have*

$$\mathit{Impl} \sqsubseteq^c_{bfr,acc} \mathit{Spec} \quad \textit{iff} \quad \mathit{Impl} \sqsubseteq_{\mathcal{F}^+_{b,fin}} \mathit{Spec}\,.$$

Proof. $\Rightarrow$: By Lemma 7.22, $\sqsubseteq^c_{bfr,acc}$ implies $\sqsubseteq_{b,acc}$ from which we conclude $bound_b$-inclusion. It remains to show $\mathcal{F}^+_{b,fin}$-inclusion. Let $(w, X, Y) \in \mathcal{F}^+_{b,fin}(\mathit{Impl})$ with $w \notin bound_b(\mathit{Impl})$. Otherwise, $w \in bound_b(\mathit{Impl}) \subseteq bound_b(\mathit{Spec})$ by $bound_b$-inclusion, and $(w, X, Y) \in \mathcal{F}^+_{b,fin}(\mathit{Spec})$ by Definition 7.12. We use net N in Fig. 6 as in the proof of Theorem 4.17. Following the argumentation in the proof of Theorem 4.17, we have that if an open net C is not a *bfr*-controller of $\mathit{Impl} \oplus N$ so it is not a *bfr*-controller of $\mathit{Spec} \oplus N$. We distinguish three cases: If $w \in bound_b(\mathit{Spec})$, then $(w, X, Y) \in \mathcal{F}^+_{b,fin}(\mathit{Spec})$ by Definition 7.12. If $wu \in bound_b(\mathit{Spec})$ with $u \in {\downarrow} X \cup {\downarrow} Y$, then $(wu, u^{-1}X, u^{-1}Y) \in \mathcal{F}^+_{b,fin}(\mathit{Spec})$ by Definition 7.12. Otherwise, we use the argumentation in the proof of Theorem 4.17 to conclude that $(w, X, Y) \in \mathcal{F}^+_{b,fin}(\mathit{Spec})$.

$\Leftarrow$: Let $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+_{b,fin}} \mathit{Spec}$. We conclude $bound_b$-inclusion by Definition 7.13(2) and L_b-, $stop_b$-, and $dead_b$-inclusion by Lemma 7.14. Now Theorem 7.7 implies $\mathit{Impl} \sqsubseteq_{bfr,acc} \mathit{Spec}$, and this, in turn, also shows that $\mathit{Impl} \sqsubseteq_{\mathcal{F}^+_{b,fin}} \mathit{Spec}$ implies $\mathit{Impl} \sqsubseteq^c_{bfr,acc} \mathit{Spec}$ with Theorem 7.21 and the definition of $\sqsubseteq^c_{bfr,acc}$. □

Example 7.5. In Example 7.2, we showed that D' *bfr*-accords with D (with the empty set of final markings) but D' does not $\mathcal{F}^+_{b,fin}$-refine D by Example 7.4. Consequently, $D' \sqsubseteq^c_{bfr,acc} D$ does not hold.

7.4 Decidability of $\mathcal{F}^+_{b,fin}$-refinement

In this section, we show that checking $\mathcal{F}^+_{b,fin}$-refinement is decidable. Checking $\mathcal{F}^+_{b,fin}$-refinement entails checking both items of Definition 7.13. The first item of Definition 7.13—that is, checking $bound_b$-inclusion—is decidable; see Sect. 6.4. To decide refinement of the b-bounded fintree failures—that is, the second item of Definition 7.13—we apply a similar proof strategy as in the case of $\mathcal{F}^+_b$-refinement. First, we show that the construction of Rensink and Vogler [29, Theorem 61] for deciding $\mathcal{F}^+$-refinement of two finite LTS can be generalized to decide $\mathcal{F}^+_{fin}$-refinement. Then we present an encoding of the b-bounded fintree failures of two open nets as finite LTS on which we decide $\mathcal{F}^+_{fin}$-refinement. That way, we can conclude decidability of $\mathcal{F}^+_{b,fin}$-refinement.

Deciding $\mathcal{F}^+_{fin}$-refinement for finite LTS We assume two finite transition systems *Impl* and *Spec* with initial states p_0 and q_0 such that $L(Impl) \subseteq L(Spec)$. Checking language inclusion is known to be decidable. We denote the transition relations with $\rightarrow_{Impl}$ and $\Rightarrow_{Impl}$.

We can view each finite transition system as a finite automaton where all states are accepting. For an automaton A with some state s (we write $s \in A$), $L_A(s)$ denotes the language of the automaton if we change the initial state to s. Observe that the arc label τ corresponds to the empty word in automata theory. We consider an automaton A with two types of accepting states, called 1-accepting and 2-accepting. With $L^i(A)$ we denote the language of A when i-accepting states are considered to be accepting states, for $i = 1, 2$. For the sake of notation, $L(A)$ denotes $L^1(A) \cup L^2(A)$. We call a state *productive*, if it lies on a path from the initial state to some accepting state—that is, if it is used by the automaton when accepting a word.

As a first step, we extend the automaton *Impl* to an automaton of automata AA by adding a family of pairs of deterministic automata (A^1_p, A^2_p), $p \in Impl$, such that for every $p \in Impl$ the language of A^1_p is the set $\Sigma^* \setminus L_{Impl}(p)$ of traces that *Impl* cannot perform from p, and the language of A^2_p is the set $\Sigma^* \setminus L_{fin}(p)$ with $L_{fin}(p) = \{w \mid p \xrightarrow{w} p' \wedge p' \in \Omega_{Impl}\}$. The following holds:

$$(w, X, Y) \in \mathcal{F}^+_{b,fin}(Impl) \text{ iff } \exists p_0 \stackrel{w}{\Longrightarrow}_{AA} p : X \subseteq L(A^1_p) \wedge Y \subseteq L(A^2_p).$$

Thus the automata A_p represent some b-bounded fintree failures $(w, X, Y) \in \mathcal{F}^+_{b,fin}(Impl)$ in the sense that there is a $p \in Impl$ with $p_0 \stackrel{w}{\Longrightarrow}_{AA} p$ and $X \subseteq L(A^1_p)$ and $Y \subseteq L(A^2_p)$; in particular, they represent all maximal b-bounded fintree failures—that is, all those $(w, X, Y) \in \mathcal{F}^+_{b,fin}(Impl)$ with maximal X and Y. Note that in a finite transition system *Impl*, there exists for each $(v, X', Y') \in \mathcal{F}^+_{b,fin}(Impl)$ a maximal $(v, X, Y) \in \mathcal{F}^+_{b,fin}(Impl)$ with $X' \subseteq X$ and $Y' \subseteq Y$.

Similar, we construct an automaton of pairs of automata for *Spec*, but this time, we additionally make *Spec* deterministic more or less by the usual powerset construction. This results in a deterministic automaton of pairs of automata BB,

which is a deterministic automaton extended with a family BB_Q, $Q \in BB$. For each state Q (being a set of states of *Spec*), BB_Q is a pair of deterministic automata (B^1_q, B^2_q), $q \in Q$ with $L(B^1_q) \subseteq \Sigma \setminus L_{Spec}(q)$ and $L(B^2_q) \subseteq \Sigma \setminus L_{fin,Spec}(q)$.

More in detail, the automaton part of BB is defined as follows: The initial state of BB is $Q_0 = \{q \mid q_0 \xrightarrow{\tau^*}_{Spec} q\}$; the transition relation is defined by $Q \xrightarrow{a}_{BB} Q'$ if $Q' = \{q' \mid \exists q \in Q : q \xrightarrow{a}_{Spec} \xrightarrow{\tau^*}_{Spec} q'\}$. We restrict BB to the nonempty states reachable from Q_0 and let each state of BB be accepting. As a consequence, all states of BB are productive and $L(BB) \subseteq L(Spec)$.

This way, $Q_0 \overset{w}{\Longrightarrow}_{BB} Q$ iff $Q = \{q \mid q_0 \overset{w}{\Longrightarrow}_{Spec} q\}$ for all $w \in \Sigma^*$ and

$$(w, X, Y) \in \mathcal{F}^+_{b,fin}(Spec) \text{ iff}$$
$$\exists Q_0 \overset{w}{\Longrightarrow}_{BB} Q, (B^1, B^2) \in BB_Q : X \subseteq L(B^1) \wedge Y \subseteq L(B^2).$$

First, we construct the following partial product automaton S, which can also be seen as the minimal simulation from AA to BB:

- $(p_0, Q_0) \in S$ is the initial state of S and all states are accepting.
- If $(p, Q) \in S, a \in \Sigma$ and $p \xrightarrow{a}_{AA} p'$, then by language inclusion and definition of BB, there is a unique $Q' \in BB$ such that $Q \xrightarrow{a}_{BB} Q'$; we add (p', Q') and the transition $(p, Q) \xrightarrow{a} (p', Q')$ to S.
- If $(p, Q) \in S$ and $p \xrightarrow{\tau}_{AA} p'$, then we add (p', Q) and the transition $(p, Q) \xrightarrow{\tau} (p', Q)$ to S (recall that BB has no τ labeled arcs).

Checking $\mathcal{F}^+_{b,fin}$-refinement entails checking whether for all $(w, X, Y) \in \mathcal{F}^+_{b,fin}(Impl)$ with $X \cup Y \neq \emptyset$, we have $(wu, u^{-1}X, u^{-1}Y) \in \mathcal{F}^+_{b,fin}(Spec)$ for some $u \in {\downarrow}(X \cup Y)$. Recall that by language inclusion we do not have to check triples $(w, \emptyset, \emptyset)$. We have to check for each $(p, Q) \in S$ and each pair (X, Y) with $X \subseteq L(A^1_p)$, $Y \subseteq L(A^2_p)$, and $X \cup Y \neq \emptyset$ that

$$\begin{aligned} \exists u \in {\downarrow}(X \cup Y), Q' \in BB, (B^1, B^2) \in BB_{Q'} : \\ Q \overset{u}{\Longrightarrow}_{BB} Q' \wedge u^{-1}X \subseteq L(B^1), u^{-1}Y \subseteq L(B^2) \end{aligned} \tag{1}$$

Let us fix (p, Q); we now show how to check (1) for all suitable (X, Y). This means that we have to compare runs in A^1_p or A^2_p (u in (1)) with runs of BB. To do this, we construct another (partial) product automaton P, similar to the previous one, but this time between the automata A^1_p, A^2_p (whose initial states we also denote by p) and BB where the initial state is changed to Q. Another difference with the case above is that, this time, we do not necessarily have $L(A^1_p) \cup L(A^2_p) \subseteq L_{BB}(Q)$—that is, BB might not be able to simulate both, A^1_p and A^2_p—but still we want to represent all of $L(A^1_p) \cup L(A^2_p)$ in order to check the inclusion in (1). Therefore, P is constructed as follows (here $*$ is a dummy element, not appearing anywhere else):

- $(p, p, Q) \in P$ is the initial state;

- $(p_1, p_2, Q') \in P$ and $p_1 \xrightarrow{a}_{A_p^1} p_1'$ or $p_2 \xrightarrow{a}_{A_p^2} p_2'$ (implying $p_1 \neq * \neq p_1'$ or $p_2 \neq * \neq p_2'$), we add state (p_1'', p_2'', Q'') and the transition $(p_1, p_2, Q') \xrightarrow{a} (p_1'', p_2'', Q'')$ where
  - p_i'' is p_i' if $p_i \xrightarrow{a}_{A_p^i} p_i'$ and $*$ otherwise (in particular if $p_i = *$) for $i = 1, 2$
  - Q'' satisfies $Q' \xrightarrow{a}_{BB} Q''$ (in particular, $Q' = *$ or is $*$ otherwise
- (p_1, p_2, Q') is i-accepting if p_i is accepting in A_p^i, $i = 1, 2$.

Because the A_p^i and BB are deterministic, P is also deterministic, and we have $L^i(P) \subseteq L(A_p^i)$ for $i = 1, 2$ by construction. We will call R a *productive subautomaton* of P, if R is obtained from P by restricting all components (in particular also the accepting states) to a subset M of the state set such that each state of R is productive in R[5]. We will show that (1) is satisfied for all suitable (X, Y) if and only if for each productive subautomaton R of P

$$\begin{aligned} \exists (p_1, p_2, Q') \in R, Q' \in BB, (B^1, B^2) \in BB_{Q'} : \\ L_R^i(p_1, p_2, Q') \subseteq L(B^i), \text{ for } i = 1, 2 \end{aligned} \tag{2}$$

Because the latter is clearly decidable, it then follows that $\mathcal{F}^+_{b,fin}$-refinement is decidable. Note that $Q' \in BB$ in (2) is equivalent to $Q' \neq *$.

So assume (1) is satisfied for all suitable (X, Y). If R is a productive subautomaton, then $L^1(R) \subseteq L^1(P) \wedge L^2(R) \subseteq L^2(P) \wedge L^1(R) \cup L^2(R) \neq \emptyset$. Hence, due to (1) and the construction of P, $\exists u \in \downarrow L^1(R) \cup \downarrow L^2(R), Q' \in BB, (B^1, B^2) \in BB_{Q'} : Q \xRightarrow{u}_{BB} Q'$ and $u^{-1}L^i(R) \subseteq L(B^i)$, for $i = 1, 2$. Then $(p, p, Q) \xRightarrow{u}_R (p_1, p_2, Q')$ for some p_1, p_2; because R is deterministic, (p_1, p_2, Q') is uniquely determined by u, and therefore $u^{-1}L^i(R) = L_R^i((p_1, p_2, Q'))$. Thus, (p_1, p_2, Q') and (B^1, B^2) are the state and automaton pair whose existence is asserted in (2).

Vice versa, assume that (2) holds for each productive subautomaton R and take some $X \subseteq L(A_p^1)$ and $Y \subseteq L(A_p^2)$ with $X \cup Y \neq \emptyset$. The set of states that are needed in P to accept the words of $X \cup Y$ (recall the construction of P) defines a productive subautomaton R with $X \subseteq L^1(R)$ and $Y \subseteq L^2(R)$. Take $(p_1, p_2, Q') \in R$ and $(B^1, B^2) \in BB_{Q'}$ that satisfies (2). Then there is some $u \in \downarrow X \cup \downarrow Y$ with $(p, p, Q) \xRightarrow{u}_P (p_1, p_2, Q')$ by choice of R and $Q \xRightarrow{u}_{BB} Q'$ by construction of P and because $Q' \in BB$. Now $u^{-1}X \subseteq u^{-1}L^1(R) = L_R^1((p_1, p_2, Q'))$ and $u^{-1}Y \subseteq u^{-1}L^2(R) = L_R^1((p_1, p_2, Q'))$ by determinism of R, and we can conclude that $u^{-1}X \subseteq L(B^1)$ and $u^{-1}Y \subseteq L(B^2)$.

Therefore, we have shown:

Proposition 7.24. *Checking $\mathcal{F}^+_{fin}$-refinement for two finite LTS is decidable.*

[5] Recall that a state of an automaton is productive if it lies on a path from the initial to some accepting state.

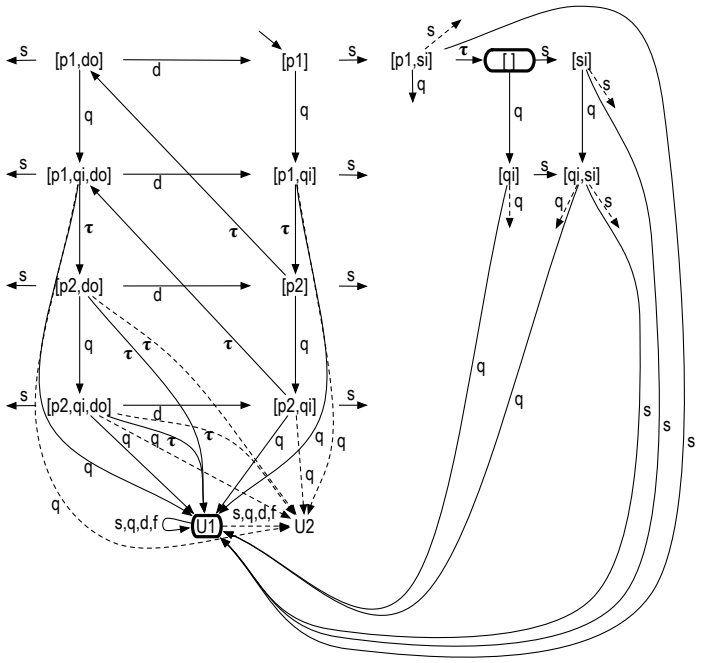


Fig. 16: Sketch of the finite automaton $U_{12}(D)$ as used in the proof of Theorem 7.28. The two final states U_1 and $[\,]$ are depicted by a thick frame. An arrow without sink sketches a transition, with sketched dashed transitions always leading to state U_2.

Reducing decision of $\mathcal{F}^+_{b,fin}$-refinement to $\mathcal{F}^+_{fin}$-refinement As we did for $\mathcal{F}^+_b$-refinement in Sect. 6.4, we prove decidability of $\mathcal{F}^+_{b,fin}$-refinement for arbitrary LTS rather than for reachability graphs of labeled nets. For an LTS S, we construct a finite LTS $U_{12}(S)$ (see Sect. 6.4). In this section we consider—in contrast to Sect. 6.4—LTS with final states and define the set of final states of $U_{12}(S)$ by the final states of S and additionally U_1.

Example 7.6. Figure 16 sketches the automaton that represents $bound_1(D)$. Parts of the automaton in Fig. 16 that can only be reached with an s-labeled transition are not shown. Likewise, the q-labeled transition leaving the state $[p_1, s^i]$ is not shown. The set of final markings of the open net D is $\{[\,]\}$; thus, $U_{12}(D)$ has final states $[\,]$ and U_1.

The next lemma, gives three observations about $U_{12}(S)$.

Lemma 7.25. *Let S be an LTS. Then the following facts hold for $U_{12}(S)$.*

1. *$w \in bound_b(S)$ iff $m_N \stackrel{w}{\Longrightarrow} U_1$ iff $m_N \stackrel{w}{\Longrightarrow} U_2$.*

2. $B\mathcal{F}^+_{b,fin}(S) \subseteq \mathcal{F}^+_{fin}(U_{12}(S))$ *(i.e., U_2 can refuse all $X \subseteq (I \cup O)^+$ and fin-refuse all $Y \subseteq (I \cup O)^*$)*
3. $\mathcal{F}^+_{fin}(U_{12}(S)) \subseteq \mathcal{F}^+_{b,fin}(S)$ *and* $(w, X, Y) \in \mathcal{F}^+_{b,fin}(S) \setminus \mathcal{F}^+_{fin}(U_{12}(S))$ *implies* $\exists u \in \downarrow (X \cup Y) \setminus X$ *such that* $wu \in bound_b(S)$

Proof. (1) follows immediately from the definition of $U_{12}(S)$, and (2) is an implication of (1). (3) The sets agree on those (w, X, Y) with $w \in bound_b(S)$ by (2). So consider $w \notin bound_b(S)$. If $(w, X, Y) \in \mathcal{F}^+_{fin}(U_{12}(S))$ due to $m_S \stackrel{w}{\Longrightarrow} m$, then we also have $m_S \stackrel{w}{\Longrightarrow} m$ in S with the same run. In $U_{12}(S)$, m could only have more traces (possibly to final states) due to runs using U_1, so it can only refuse and fin-refuse less. Thus, $(w, X, Y) \in \mathcal{F}^+_{fin}(S)$ and inclusion follows.

If $(w, X, Y) \in \mathcal{F}^+_{b,fin}(S)$ due to m, but $(w, X, Y) \notin \mathcal{F}^+_{fin}(U_{12}(S))$, then this must be due to a run from m that touches U_1; assume this happens for the first time after $u \neq \varepsilon$; that is, we have $m_N \stackrel{w}{\Longrightarrow} m \stackrel{u}{\Longrightarrow} U_1 \stackrel{u'}{\Longrightarrow}$ with $uu' \in X \cup Y$. Thus, $wu \in bound_b(S)$ and $u \in \downarrow (X \cup Y)$. Because $m_N \stackrel{w}{\Longrightarrow} m \stackrel{u}{\Longrightarrow}$ also in S, we further have $u \notin X$. □

The following lemma gives another prerequisite for the decidability proof.

Lemma 7.26. *For $X, Y \in \mathcal{P}(\Sigma^*)$, let $x \in \downarrow (X \cup Y) \cup \{\varepsilon\}$, $X' = x^{-1}X$ and $Y' = x^{-1}Y$. Then*

1. *$u \in \downarrow (X' \cup Y')$ implies $xu \in \downarrow (X \cup Y)$*
2. *$u \notin X'$ implies $xu \notin X$*

Proof. (1) Due to $uu' \in X' \cup Y'$ and $xuu' \in X \cup Y$.
(2) Suppose $xu \in Y$. Then $u \in X$ and we have a contradiction. □

With the next lemma, we show that deciding $\mathcal{F}^+_{b,fin}$-refinement for two open nets *Impl* and *Spec* reduces to check $\mathcal{F}^+_{fin}$-refinement of their finite automata $U_{12}(Spec)$ and $U_{12}(Impl)$.

Lemma 7.27. *For two open nets Impl and Spec such that $bound_b(Impl) \subseteq bound_b(Spec)$, we have*

$$Impl \sqsubseteq_{\mathcal{F}^+_{b,fin}} Spec \text{ iff } U_{12}(Impl) \sqsubseteq_{\mathcal{F}^+_{fin}} U_{12}(Spec)\,.$$

Proof. $\Rightarrow$: Each $(v, X, Y) \in \mathcal{F}^+_{fin}(U_{12}(Impl)) \subseteq \mathcal{F}^+_{b,fin}(Impl)$ (by Lemma 7.25(3)) is dominated by some $(w, X', Y') \in \mathcal{F}^+_{b,fin}(Spec)$ due to some $x \in \downarrow (X \cup Y) \cup \{\varepsilon\}$ i.e., $X' = x^{-1}X$, $Y' = x^{-1}Y$, $w = vx$. If $(w, X', Y') \in \mathcal{F}^+_{fin}(U_{12}(Impl))$, we are done. So assume otherwise and consider $u \in \downarrow (X' \cup Y') \setminus X'$ according to Lemma 7.25(3). Then $xu \in \downarrow (X \cup Y) \setminus X$ by Lemma 7.26 and $(vxu, (xu)^{-1}X, (xu)^{-1}Y) \in \mathcal{F}^+_{fin}(U_{12}(Spec))$ because $vxu \in bound_b(Spec)$ and $\varepsilon \notin (xu)^{-1}X$. Hence, (v, X, Y) is also covered in this case.

$\Leftarrow$: Each $(w, X, Y) \in \mathcal{F}^+_{fin}(U_{12}(Impl))$ is dominated by $\mathcal{F}^+_{fin}(U_{12}(Spec)) \subseteq \mathcal{F}^+_{b,fin}(Spec)$ by Lemma 7.25(3). So consider $(w, X, Y) \in \mathcal{F}^+_{b,fin}(Impl) \setminus$

$\mathcal{F}^+_{fin}(U_{12}(Impl))$ and respectively $u \in \downarrow (X \cup Y) \setminus X$ according to Lemma 7.25(3). Then, $(wu, u^{-1}X, u^{-1}Y) \in \mathcal{F}^+_{b,fin}(Spec)$ because $wu \in bound_b(Impl) \subseteq bound_b(Spec)$ and $\varepsilon \notin u^{-1}X$. Thus, (w, X, Y) is also dominated in this case. □

With Lemma 7.27 and Proposition 7.24, we have shown:

Theorem 7.28 ($\mathcal{F}^+_{b,fin}$-refinement is decidable). *For two interface-equivalent open nets Impl and Spec, checking whether Impl $\sqsubseteq_{\mathcal{F}^+_{b,fin}}$ Spec is decidable.*

8 Related Work

The idea of responsiveness for finite state open systems with final states has been coined by Wolf in [37]: An open net N is responsive if $inner(N)$ is b-bounded and from every reachable marking we can reach either a final marking or a marking that enables a transition with an output place in its postset [37]. In other words, Wolf defines responsiveness for single open nets and considers only such responsive nets; this guarantees a stricter form of our respective responsiveness variant. More generally, we also deal with open nets that are responsive in some open net compositions but not in others. Lohmann and Wolf [20] present a more efficient decision procedure for the responsiveness in [37], but on an automaton model. Responsiveness in [37,20] is mainly motivated by algorithmic considerations for deciding the respective accordance, but without characterizing the latter semantically or studying compositionality. Müller [24] presents an asymmetrical definition from the point of view of one individual open system in a composition. Our notion of responsiveness leads to precongruences, where the related equivalence is similar to P-deadlock equivalence in [33]. Desai et al. [11] define responsiveness for bounded message queues. Their notion of responsiveness is stricter than ours because it additionally requires that no message in any channel is ignored forever.

In other work, the term responsiveness refers to different properties: Reed et al. [28] aim at excluding certain deadlocks, whereas responsiveness in our setting refers to the ability to communicate. The works [16,1,13] consider with the π-calculus a more expressive model than open nets but in the setting of synchronous communication, whereas we consider asynchronous communication. Moreover, responsiveness in [1,13] and lock-freedom in [16] guarantee that communication over a certain channel is eventually possible. In contrast, our notion of responsiveness requires that communication over some channel is always possible. Kobayashi [16] defines responsiveness over the infinite runs of the system, thereby using a strong fairness for the channel synchronization. Moreover, the language considered in [16] does not support choices. Acciai and Boreale [1] use a type system and reduction rules different from Kobayashi, and they give an example of a responsive process that cannot be expressed in [16]. Gamboni and Ravara [13] use a variant of the π-calculus that is more expressive than the one in [16]: Choices are part of the language and it is also possible to express how many times a communication over a certain channel should take place. In addition to responsiveness in [1] (called activeness in [13]), Gamboni and Ravara

require that whenever communication takes place over a channel, then the respective processes conform to to a specified protocol. The latter property is called responsiveness in [13]. Recently, Padovani [26] takes up lock-freedom from [16]. He defines a behavioral type system using asynchronously communicating session types and considers the progress property. Progress is a stricter notion than bounded final responsiveness as it (like responsiveness in [11]) additionally requires that no message in any channel is ignored forever.

Trace-based semantics like ours (in particular in Sect. 5) where the language is flooded with error traces go back to the work of Dill [12]. Errors in [12] arise from communication mismatches and are similar to our bound-violators. In contrast to our asynchronous (i.e., buffered) setting, Dill considers a synchronous (i.e., an unbuffered) setting. Dill's semantics can be seen as declarative whereas interface automata, as defined by Alfaro and Henzinger in [10], take up the same ideas on an operational (i.e., automaton) model. Dill's refinement relations are trace inclusions like our characterizations of the four responsiveness preorders. In contrast, refinement of interface automata is characterized by an alternating simulation relation similar to the refinement of modal transition systems [17]. Chen et al. [8] (long version in [9]) present more recent work that is based on languages. They consider with quiescent traces a kind of stop traces. The precongruence defined in [8] is based on traces but in a synchronous setting. In contrast, our precongruences are variants of the should testing preorder [29]. Moreover, our notion of divergence (i.e., "infinite internal chatter") is different from the one in [9]: If two open systems indefinitely interact with each other, then they are responsive and, hence, we do not treat such trace as problematic, but Chen et al. [8] do. The reason is that, intuitively, we assume a stronger fairness. Common for the synchronous setting of Dill [12], Alfaro and Henzinger [10], and Chen et al. [8] is that they all have to apply some kind of output pruning: In the composition of two open systems, if a sequence of output transitions leads to an error state, these transitions and the states involved have to be removed.

Compared to our previous work on deadlock freedom in [32], finer trace sets are required to characterize the preorders based on responsiveness. While traces are adequate for the precongruence dealing with deadlock freedom [32], they do not suffice to characterize the coarsest precongruence for responsiveness, and we had to use some kind of failures instead. In unpublished work, we showed undecidability for the unbounded preorders and precongruences.

The results in Sect. 5 on boundedness without responsiveness are mainly extracted from the bounded variant of deadlock freedom in [32]. Brand and Zafiropolu investigated channel boundedness in [4]. Haddad et al. [15] study channel properties for asynchronously communicating Petri nets, present compositionality results, and show decidability. The considered properties are concerned with the consumption of messages sent on a channel, such as upper and lower bounds for pending messages, in the presence and absence of fairness. In contrast, boundedness in our setting only restricts the number of pending messages and is, thus, a more abstract and fundamental concept. We do not require that pending messages have to be consumed. Moreover, we assume a strong no-

tion of fairness by considering a state as satisfactory where it is possible to reach a final state or a state where communication is possible.

The notion of uncoverable states has been observed as the set of certain conflicts by Malik et al. [21] for a stronger correctness criterion than responsiveness. A similar phenomenon occurs in the safe-must preorder [3], where a process and its observer must reach a success state before reaching a catastrophic (i.e., diverging) one.

9 Conclusion

We studied an accordance preorder describing whether an open system can safely be replaced by another open system, thereby guaranteeing responsiveness of the overall system. The latter guarantees deadlock freedom and the permanent possibility to mutually communicate. Responsiveness can be seen as a minimal correctness criterion for open systems. We investigated two dimensions of responsiveness, where we can additionally consider final states or require boundedness of the composition due to maintaining a previously known message bound. Altogether, this resulted in four variants of the accordance preorder.

For each variant of accordance, we presented a trace-based characterization. In its basic form, the semantics consist of a set collecting completed traces. In the presence of final states, an additional set is needed to distinguish successfully and unsuccessfully completed traces. To deal with boundedness, we had to add the language and a set of uncoverable traces collecting catastrophic traces that cannot be used reliably.

We showed that none of the four accordance preorders is a precongruence and characterized the coarsest precongruence that is contained in the respective preorder. For basic responsiveness, this precongruence is the should testing preorder [29]. In the presence of final states, it is the should testing preorder extended with traces that do not lead to a final marking. For boundedness, we had to extend the should testing preorder by information about bound violations. In the unbounded setting, we showed in unpublished work that neither the preorders nor the precongruences are decidable. This gives an important motivation to deal with the bounded setting. For the latter setting, we proved decidability of the precongruences.

It is future work to study the relation of our semantics and the compact representation of all controllers in [20]. In particular, we are interested in using such representations to decide the coarsest precongruences. Another issue is the minimal requirement *weak termination* (e.g., [21,23]): Reaching a final state should always be possible. This criterion is very close to the idea of should testing, but it is not clear how to characterize the respective accordance (which is a precongruence itself). In contrast, we characterized precongruences related to responsiveness with semantical ideas that also worked for should testing.

References

1. Acciai, L., Boreale, M.: Responsiveness in process calculi. Theor. Comp. Sci. 409(1), 59–93 (2008)
2. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38(4), 393 – 422 (2002)
3. Boreale, M., De Nicola, R., Pugliese, R.: Basic observables for processes. Inf. Comput. 149(1), 77–98 (1999)
4. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM 30(2), 323–342 (1983)
5. Bravetti, M., Zavattaro, G.: Contract based multi-party service composition. In: FSEN 2007. LNCS, vol. 4767, pp. 207–222 (2007)
6. Brinksma, E., Rensink, A., Vogler, W.: Fair testing. In: CONCUR 1995. LNCS, vol. 962, pp. 313–327. Springer (1995)
7. Brookes, S.D., Hoare, C.A.R., Roscoe, A.W.: A theory of communicating sequential processes. J. ACM 31(3), 560–599 (1984)
8. Chen, T., Chilton, C., Jonsson, B., Kwiatkowska, M.Z.: A compositional specification theory for component behaviours. In: ESOP. LNCS, vol. 7211, pp. 148–168. Springer (2012)
9. Chilton, C., Jonsson, B., Kwiatkowska, M.: An algebraic theory of interface automata. Tech. Rep. RR-13-02, DCS (2013)
10. de Alfaro, L., Henzinger, T.A.: Interface automata. In: ESEC / SIGSOFT FSE 2001. pp. 109–120 (2001)
11. Desai, A., Gupta, V., Jackson, E., Qadeer, S., Rajamani, S.K., Zufferey, D.: P: safe asynchronous event-driven programming. In: PLDI 2013. pp. 321–332. ACM (2013)
12. Dill, D.L.: Trace theory for automatic hierarchical verification of speed-independent circuits. ACM distinguished dissertations, MIT Press (1989)
13. Gamboni, M., Ravara, A.: Responsive choice in mobile processes. In: TGC, LNCS, vol. 6084, pp. 135–152. Springer (2010)
14. Glabbeek, R.J.v.: The coarsest precongruences respecting safety and liveness properties. In: TCS 2010. IFIP, vol. 323, pp. 32–52. Springer (2010)
15. Haddad, S., Hennicker, R., Møller, M.H.: Channel properties of asynchronously composed petri nets. In: Petri Nets. LNCS, vol. 7927, pp. 369–388. Springer (2013)
16. Kobayashi, N.: A type system for lock-free processes. Information and Computation 177(2), 122–159 (2002)
17. Larsen, K.G.: Modal specifications. In: AVMFSS 1989. LNCS, vol. 407, pp. 232–246. Springer (1990)
18. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: ICATPN 2007. LNCS, vol. 4546, pp. 321–341. Springer (2007)
19. Lohmann, N., Verbeek, E., Dijkman, R.M.: Petri net transformations for business processes - a survey. In: ToPNoC II. pp. 46–63. LNCS 5460, Springer (2009)
20. Lohmann, N., Wolf, K.: Compact representations and efficient algorithms for operating guidelines. Fundam. Inform. 107, 1–19 (2011)
21. Malik, R., Streader, D., Reeves, S.: Conflicts and fair testing. Journal of Foundations of Computer Science 17(4), 797–813 (2006)
22. Milner, R.: Communication and Concurrency. Prentice-Hall, Inc. (1989)
23. Mooij, A.J., Stahl, C., Voorhoeve, M.: Relating fair testing and accordance for service replaceability. J. Log. Algebr. Program. 79(3-5), 233–244 (2010)

24. Müller, R.: On the notion of deadlocks in open nets. In: Schwarick, M., Heiner, M. (eds.) AWPN 2010. CEUR Workshop Proceedings, vol. 643, pp. 130–135. CEUR-WS.org (2010)
25. Natarajan, V., Cleaveland, R.: Divergence and fair testing. In: ICALP. LNCS, vol. 944, pp. 648–659. Springer (1995)
26. Padovani, L.: From lock freedom to progress using session types. To appear for PLACES 2013
27. Papazoglou, M.P.: Web Services: Principles and Technology. Pearson (2007)
28. Reed, J.N., Roscoe, A.W., Sinclair, J.E.: Responsiveness and stable revivals. Formal Asp. Comput. 19(3), 303–319 (2007)
29. Rensink, A., Vogler, W.: Fair testing. Inf. Comput. 205(2), 125–198 (2007)
30. Stahl, C., Massuthe, P., Bretschneider, J.: Deciding substitutability of services with operating guidelines. In: ToPNoC II. pp. 172–191. LNCS 5460, Springer (2009)
31. Stahl, C., Vogler, W.: A trace-based view on operating guidelines. In: FOSSACS 2011. LNCS, vol. 6604, pp. 411–425. Springer (2011)
32. Stahl, C., Vogler, W.: A trace-based service semantics guaranteeing deadlock freedom. Acta Inf. 49(2), 69–103 (2012)
33. Vogler, W.: Modular Construction and Partial Order Semantics of Petri Nets, LNCS, vol. 625. Springer (1992)
34. Vogler, W., Stahl, C., Müller, R.: Trace- and failure-based semantics for bounded responsiveness. Accepted for FOCLASA 2013
35. Vogler, W., Stahl, C., Müller, R.: A trace-based semantics for responsiveness. In: ACSD 2012. pp. 42–51. IEEE (2012)
36. Voorhoeve, M., Mauw, S.: Impossible futures and determinism. Inf. Process. Lett. 80(1), 51–58 (2001)
37. Wolf, K.: Does my service have partners? In: ToPNoC II. pp. 152–171. LNCS 5460, Springer (2009)