Undecidability of accordance for open systems with unbounded message queues

Richard Müller[a,b,*], Christian Stahl[b], Walter Vogler[c]

[a] *Institut für Informatik, Humboldt-Universität zu Berlin, Germany*
[b] *Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, The Netherlands*
[c] *Institut für Informatik, Universität Augsburg, Germany*

Abstract

We study asynchronously communicating open systems modeled as Petri nets with an interface. An accordance preorder describes when one open system can be safely replaced by another open system without affecting some behavioral property of the overall system. Although accordance is decidable for several behavioral properties if we assume a previously known bound on the maximal number of pending messages, we show that it is not decidable without this assumption.

Keywords: Petri nets, open nets, accordance preorder, theory of computation

1. Introduction

Today's software systems are complex distributed systems that are composed of less complex open systems. In this paper, we focus on stateful open systems that have a well-defined interface and communicate with each other via asynchronous message passing. Service-oriented systems like Web-service applications [1] 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 an open system is replaced by another one—for example, when new features have been implemented or bugs have been fixed. To describe what replacements are acceptable a refinement notion is required, which can be formalized as an accordance preorder. An accordance preorder indicates whether we can safely replace an open system by another one without affecting some relevant behavioral property of the overall system.

Here, we consider open systems on an abstract level—for example, abstracting from message contents—and model them with open (Petri) nets. Decision procedures for accordance exist for deadlock freedom [3] and responsiveness [4] (i.e., the perpetual possibility to communicate), if we assume a previously known bound on the maximal number of pending messages between the open systems. This bound has to be determined beforehand by, for example, static analysis of the system's underlying middleware or of the communication behavior of an open system. A natural question is whether this previously known bound is necessary. In this paper, we give a negative answer: We prove that accordance is undecidable for deadlock freedom and responsiveness—both with and without final states—and weak termination [5] (i.e., the perpetual possibility to reach a final state).

We continue with some background information on Petri nets and accordance in Sect. 2. In Sect. 3, we prove the undecidability of accordance for deadlock freedom. We lift this result to accordance for responsiveness in Sect. 4. Section 5 contains the undecidability result of accordance for weak termination, and Section 6 finishes with a discussion of related work.

2. Preliminaries

In this section, our presentation largely follows [6, 7]. 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 $\mathbb{N}$ ($\mathbb{N}_+$) denote the natural numbers (excluding 0). In this paper, we use place/transition Petri nets extended with a set of final markings and either transition labels or interface places.

Definition 1 (net)**.** *A* net $N = (P, T, F, m_N, \Omega)$ *consists of a finite set* P *of* places, *a finite 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 multiset* $m : P \to \mathbb{N}$, *and a set* Ω *of* final markings.

Where needed (Definitions 4,5), we implicitly extend a marking m *to additional places, for which* m *returns 0.*

Introducing a net N also implicitly introduces its components P, T, F, m_N, and Ω—and similarly for nets N_1, N_2.

Definition 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* input actions Σ_{in} *and*

*Corresponding author
Email addresses: richard.mueller@informatik.hu-berlin.de (Richard Müller), c.stahl@tue.nl (Christian Stahl), vogler@informatik.uni-augsburg.de (Walter Vogler)

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* action-equivalent *if they have the same sets of input and of 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. A marking is a distribution of tokens over the places. Graphically, a black dot represents a token. Transition labels are written into the respective boxes.

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. 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 the marking $m' = m - {}^\bullet t + t^\bullet$ (here, we interpret the pre- and the postset as multisets). The firing of t is denoted by $m \xrightarrow{t} m'$; that is, t is enabled at m and firing it results in m'. For a transition sequence $v = t_1 \ldots t_{k-1}$, we write $m_1 \xrightarrow{v} m_k$ when $m_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{k-1}} m_k$ and refer to it as a *run* of N. A marking m' is *reachable from* a marking m if there exists a (possibly empty) run v with $m \xrightarrow{v} m'$. A marking m' is *reachable* if it is reachable from the initial marking. In the case of labeled nets, we lift runs to actions: 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 $m_1 \xRightarrow{w} m_k$, and we refer to w as a *trace* if $m_1 = m_N$. The *language* $L(N)$ is the set of all traces of N.

For two action-equivalent labeled nets N_1 and N_2, a relation $\varrho \subseteq \mathbb{N}^{P_1} \times \mathbb{N}^{P_2}$ is a *bisimulation* [8] if for all $(m_1, m_2) \in \varrho$: (1) if $m_1 \xrightarrow{t_1} m_1'$ in N_1, there exist t_2 and m_2' such that $m_2 \xrightarrow{t_2} m_2'$ in N_2, $l_1(t_1) = l_2(t_2)$, and $(m_1', m_2') \in \varrho$; and (2) if $m_2 \xrightarrow{t_2} m_2'$ in N_2, there exist t_1 and m_1' such that $m_1 \xrightarrow{t_1} m_1'$ in N_1, $l_1(t_1) = l_2(t_2)$, and $(m_1', m_2') \in \varrho$. The labeled nets N_1 and N_2 are *bisimilar* if there exists a bisimulation relating their initial markings m_{N_1} and m_{N_2}.

As system model, we consider open nets. An open net extends a net by an asynchronous interface consisting of two disjoint sets of input and output places, which correspond to input and output channels. 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.

Definition 3 (open net)**.** *An* open net *N is a tuple $(P, T, F, m_N, I, O, \Omega)$ where $(P \uplus I \uplus O, T, F, m_N, \Omega)$ is a net, $m_N(p) = 0 = m(p)$ for all $p \in I \uplus O$ and $m \in \Omega$, ${}^\bullet p = \emptyset$ for all* input places *$p \in I$, $p^\bullet = \emptyset$ for all* output places *$p \in O$, and set $I = \emptyset$ if and only if set $O = \emptyset$.*

If $I = O = \emptyset$, then N is a closed net*. Two open nets are* interface-equivalent *if they have the same sets of input and of output places.*

Graphically, we represent an open net like a net with a dashed frame around it. The interface places are positioned on the frame.

For the composition of open nets, we assume that the sets of transitions are 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 such open nets by merging shared interface places and turning them into internal places. The definition of *composable* thereby guarantees that an open net composition is again an open net (which is possibly closed).

Definition 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 such open nets 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 $\Omega = \{m_1 + m_2 \mid m_1 \in \Omega_1, m_2 \in \Omega_2\}$.*

To give an open net N a trace-based semantics, we consider its environment $env(N)$, which we define similarly to Vogler [9]. 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). Intuitively, one can understand the construction as translating the asynchronous interface of N into a buffered synchronous interface (with unbounded buffers) described by the transition labels of $env(N)$.

Definition 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*

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

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 $L(N) = L(env(N))$.

In this paper, we consider five behavioral properties on the closed composition of two open nets: deadlock freedom with and without final markings, responsiveness with and without final markings, and weak termination.

Definition 6 (behavioral properties)**.** *Let N_1 and N_2 be composable open nets. A marking m of $N_1 \oplus N_2$ is* dead *if no transition is enabled at m, and* f-dead *if additionally $m \notin \Omega_{N_1 \oplus N_2}$. Marking m is* responsive *if we can reach from m a marking that enables a transition t with $t^\bullet \cap (O_1 \uplus O_2) \neq \emptyset$; it is* weakly terminating *if we can reach a final marking of $N_1 \oplus N_2$ from m; and m is* f-responsive *if m is responsive or weakly terminating.*

The open nets N_1 and N_2 together are deadlock free *(*f-deadlock free*) if their composition $N_1 \oplus N_2$ is a closed net and no reachable marking of $N_1 \oplus N_2$ is dead (f-dead). N_1 and N_2 are* responsive *(*f-responsive*,* weakly terminating*) if their composition $N_1 \oplus N_2$ is a closed net and every reachable marking of $N_1 \oplus N_2$ is responsive (f-responsive, weakly terminating).*

Based on a behavioral property, we define a controller C of an open net N such that N and C have that property.

Definition 7 (controller)**.** *An open net C is a* df-controller *(fdf-, r-, fr-, wt-controller) of an open net N if N and C are deadlock free (f-deadlock free, responsive, f-responsive, weakly terminating).*

If the controllers of an open net are a superset of the controllers of another open net, then the first open net is a refinement of the second; intuitively, we can safely replace the second open system by the first one without affecting the behavioral property of the overall system. We refer to the resulting refinement relation as *accordance*, which gives a necessary requirement for a refinement. As the accordance preorder for (f-)responsiveness is not compositional [6], we also define the coarsest precongruence contained in the respective preorder.

Definition 8 (accordance)**.** *Let $x \in \{df, fdf, r, fr, wt\}$. For interface-equivalent open nets Impl and Spec, Impl* x-accords *with Spec if for all open nets C the following holds: If C is an x-controller of Spec, then C is an x-controller of Impl. Let $\sqsubseteq^c_{r,acc}$ ($\sqsubseteq^c_{fr,acc}$) denote the coarsest precongruence contained in r-accordance (fr-accordance).*

3. Undecidability of *df*- and *fdf*-accordance

We prove df- and fdf-accordance to be undecidable by reducing both to the halting problem of Minsky's counter machines [10]. For the reduction, we use our trace-based characterization of df- and fdf-accordance [7], which demands specific language inclusions.

Definition 9 (*stopdead*-semantics for deadlock freedom)**.** *Let N be a labeled net. A marking m of N is a* stop except for inputs *if for all $t \in T$ with $m \xrightarrow{t}$ holds: $l(t) \in \Sigma_{in}$; it is* dead except for inputs *if additionally $m \notin \Omega$.*

The stopdead*-semantics of N is defined by the sets of traces $stop(N) = \{w \mid m_N \xRightarrow{w} m \wedge m$ is a stop except for inputs$\}$ and $dead(N) = \{w \mid m_N \xRightarrow{w} m \wedge m$ is dead except for inputs$\}$.*

Theorem 10 (df- and fdf-accordance characterization [7])**.** *For two interface-equivalent open nets Impl and Spec, the following holds: (1) Impl df-accords with Spec iff $stop(Impl) \subseteq stop(Spec)$. (2) Impl fdf-accords with Spec iff $stop(Impl) \subseteq stop(Spec)$ and $dead(Impl)) \subseteq dead(Spec)$.*

We define a counter machine as in [10].

Definition 11 (counter machine)**.** *Let $n, m \in \mathbb{N}_+$. An m-*counter machine *C with nonnegative counters $c_1, \ldots, c_m$ is a program consisting of n commands*

$$1 : CMD_1; 2 : CMD_2; \ldots ; n : CMD_n$$

where CMD_n is a $HALT$-command and the commands $CMD_1, \ldots, CMD_{n-1}$ are of the following two types (where $1 \leq k, k_1, k_2 \leq n, 1 \leq j \leq m$):

Type 1: $c_j := c_j + 1;$ *goto* k

Type 2: *if* $c_j = 0$ *then goto* k_1 *else* $(c_j := c_j - 1;$ *goto* $k_2)$

Define the set $BS(C)$ of branching states *of C as $BS(C) = \{i \in \mathbb{N}_+ \mid CMD_i$ is of type 2$\}$.*

As a running example, consider the counter machine ADD in Alg. 1. ADD has two counters c_1 and c_2, and consists of three commands: one of each type, and the $HALT$-command. It expects two given integers x_1 and x_2 as inputs, and returns their sum $x_1 + x_2$ stored in the counter c_2. The branching states of ADD are $BS(ADD) = \{1\}$, and obviously ADD halts on any inputs.

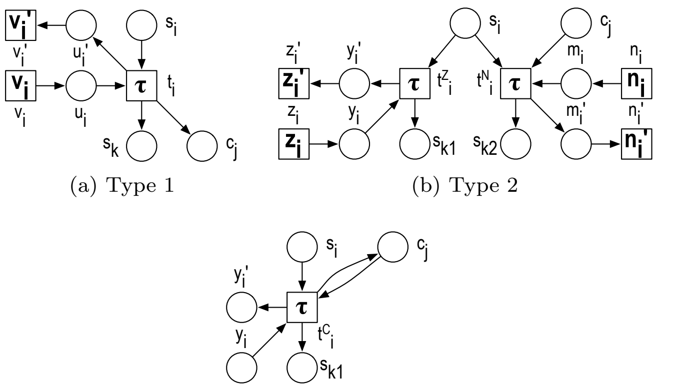
input : An integer x_1 stored in c_1, an integer x_2 stored in c_2
output: The integer $x_1 + x_2$ stored in c_2

1 if $c_1 = 0$ then goto 3 else ($c_1 := c_1 - 1$; goto 2) ;
2 $c_2 := c_2 + 1$; goto 1 ;
3 HALT

Algorithm 1: The 2-counter machine ADD for adding two integers x_1 and x_2.

We describe three labeled net patterns—one pattern for each CMD-type and an auxiliary notion of a "definitely cheating" pattern—which we use to simulate a counter machine. These patterns are an extension of the "Jančar-Pattern" [11]. For each transition t of the original pattern, we add two transitions and two places controlling t's firing. In addition, we shift the label from t to the newly introduced transitions, and label t with τ. The patterns are illustrated in Fig. 1.

Definition 12 (basic net)**.** *Let C be a counter machine with m counters and n commands. The* basic net *$net(C)$ of C is a labeled net constructed as follows (assuming $1 \leq k, k_1, k_2 \leq n, 1 \leq j \leq m$):*



(a) Type 1

(b) Type 2

(c) dc-pattern

Figure 1: Constructions of $net(C)$ and dc-pattern.

1. *Let $c_1, \ldots, c_m$ (the counter part) and $s_1, \ldots, s_n$ (the state part) be places of $net(C)$.*
2. *For $i = 1, \ldots, n-1$ add new transitions and arcs depending on the type of CMD_i:*

   type 1: *Add places u_i, u_i', transitions t_i, v_i, v_i', and arcs (v_i, u_i), (u_i, t_i), (t_i, u_i'), (u_i', v_i'), (s_i, t_i), (t_i, s_k), and (t_i, c_j). For the labeling, we set $l(v_i) = v_i$, $l(v_i') = v_i'$, and $l(t_i) = \tau$.*

   type 2: *Add places y_i, y_i', m_i, m_i', transitions t_i^Z, z_i, z_i' (to simulate the case in which counter c_j is zero) and t_i^N, n_i, n_i' (to simulate the case in which counter c_j is not empty), and arcs (z_i, y_i), (y_i, t_i^Z), (t_i^Z, y_i'), (y_i', z_i'), (n_i, m_i), (m_i, t_i^N), (t_i^N, m_i'), (m_i', n_i'), (s_i, t_i^Z), (t_i^Z, s_{k_1}), (s_i, t_i^N), (c_j, t_i^N), and (t_i^N, s_{k_2}). For the labeling, we set $l(z_i) = z_i$, $l(z_i') = z_i'$, $l(n_i) = n_i$, $l(n_i') = n_i'$, and $l(t_i^Z) = l(t_i^N) = \tau$.*
3. *Let the initial marking put just one token on s_1, and let $\emptyset$ be the set of final markings of $net(C)$.*
4. *Let every unprimed transition label of $net(C)$ (other than τ) be an input action, and let every primed transition label of $net(C)$ be an output action.*

*Adding a dc-*pattern *(dc for "definitely cheating") to $net(C)$ for $i \in BS(C)$ means adding a τ-labeled transition t_i^C (a dc-*transition*) and arcs $(y_i, t_i^C), (t_i^C, y_i')$, (s_i, t_i^C), (t_i^C, s_{k_1}), (c_j, t_i^C), (t_i^C, c_j). (Note that t_i^C is a copy of t_i^Z with additional arcs to/from c_j.)*

For the counter machine *ADD* from Alg. 1, Fig. 2 depicts the basic net $net(ADD)$. It consists of one pattern of type 1 (transitions t_2, v_2, v_2') and one pattern of type 2 (transitions t_1^N, t_1^Z, n_1, n_1', z_1, z_1'). The counters c_1 and c_2 are modeled by the places c_1 and c_2, and the current state of *ADD* is modeled by marking one of the places s_1, s_2, s_3. The input actions of $net(ADD)$ are $\{n_1, z_1, v_2\}$, and the output actions are $\{n_1', z_1', v_2'\}$.

For any counter machine with counters $c_1, \ldots, c_m$ and for any input values $x_1, \ldots, x_m$, we can "simulate" C with

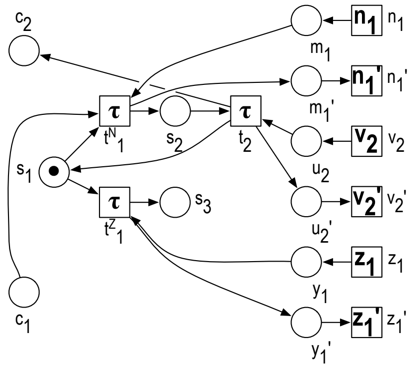


Figure 2: The basic net $net(ADD)$ of *ADD* from Alg. 1.

$net(C)$ by adding x_j tokens to the initial marking of place c_j ($1 \leq j \leq m$). However, it is possible to "cheat" in the pattern of type 2 (see Fig. 1b), i.e., transition t_i^Z fires although the respective place c_j is not empty. Also notice that firing a dc-transition has the same effect as firing the respective transition t_i^Z.

The construction of $net(C)$ applies to any counter machine, but we will consider a 2-counter machine C in the following, because already for two counters the halting problem is undecidable [10].

Theorem 13 (halting problem [10])**.** *It is undecidable whether a given 2-counter machine halts on given inputs.*

The following lemma relates the halting problem for 2-counter machines to the inclusion of the *stop*-languages of two constructed labeled nets. We follow the proof strategy from [11]: For a 2-counter machine C and given input values x_1 and x_2, we construct two labeled nets N_1 and N_2 which are modifications of $net(C)$ simulating C. The construction of N_1 and N_2 ensures that the only way to exhibit the non-inclusion is to simulate C without cheating and to terminate—which is possible if and only if C halts for x_1 and x_2.

Lemma 14. *Let C be a 2-counter machine and $x_1, x_2 \in \mathbb{N}$. We can construct two action-equivalent labeled nets N_1 and N_2 (as modifications of $net(C)$) such that the following conditions are equivalent:*

1. *C does not halt for the given inputs x_1 and x_2.*
2. *N_1 and N_2 are bisimilar.*
3. *$stop(N_1) \subseteq stop(N_2)$.*

Proof. We construct N_1 and N_2 from $net(C)$ and the input values x_1 and x_2 in four steps:

1. Take $net(C)$ and extend its initial marking by x_1 tokens in c_1 and x_2 tokens in c_2.
2. Add places p, p', e, transitions t_p, $t_{p'}$, t_e, f, and arcs (p, t_p), (t_p, p), $(p', t_{p'})$, $(t_{p'}, p')$, (p, t_e), (s_n, t_e), (t_e, e), and (e, f). Label the transitions t_p, $t_{p'}$, and t_e with τ, and f with f. Figure 3a sketches steps one and two for *ADD* with inputs $x_1 := 1$ and $x_2 := 1$.

3. For each branching state $i \in BS(C)$ that checks counter c_j, add two dc-patterns: the τ-labeled transitions t_i^C, $t_i^{C'}$, and the arcs (s_i, t_i^C), $(s_i, t_i^{C'})$, (t_i^C, s_{k_1}), $(t_i^{C'}, s_{k_1})$, $(y_i, t_i^C), (y_i, t_i^{C'}), (t_i^C, y_i'), (t_i^{C'}, y_i')$ (i.e, detecting cheating on the zero-branch), (c_j, t_i^C), (t_i^C, c_j), $(c_j, t_i^{C'}), (t_i^{C'}, c_j)$ (i.e., cheating means c_j is not empty), and $(p, t_i^C), (t_i^C, p')$, $(p', t_i^{C'}), (t_i^{C'}, p)$ (i.e., detecting cheating means switching the token between p and p'). Figure 3b (ignoring the token on place p) sketches this step for ADD and $BS(ADD) = \{1\}$.
4. Take two copies of the arising net. In one copy, put one token in p yielding the labeled net N_1. In the other, put one token in p' yielding the labeled net N_2. Figures 3b and 3c indicate this for ADD, if we ignore the dashed frame.

In every reachable marking, the places p and p' together hold at most one token. As long as place p or p' remains marked, the corresponding marking is not a stop except for inputs due to t_p and $t_{p'}$. The only way to reach a stop except for inputs is to have one token on p and fire t_e and f.

(1) implies (2): Assume C does not halt for inputs x_1 and x_2. Let D be the set of all pairs (m, m) of equal markings m of N_1 and N_2. Let M be the set of all pairs (m_1, m_2) such that m_1 and m_2 are reachable by the same correct run in N_1 and N_2, respectively. A run is correct if it simulates C without cheating, i.e., no dc-transition fires, and transition t_i^Z (for $i \in BS(C)$) fires only if the respective place c_j is empty. We show that $D \uplus M$ is a bisimulation; thus, N_1 and N_2 are bisimilar as $(m_{N_1}, m_{N_2}) \in M$ by the construction of N_1 and N_2.

So consider a pair $(m_1, m_2) \in M$. As m_1 and m_2 is reached by the same correct run σ in N_1 and N_2, respectively, m_1 and m_2 differ only in the places p and p', i.e., we have $m_1(p) = 1$, $m_1(p') = 0$, and $m_2(p) = 0$, $m_2(p') = 1$ w.l.o.g. Thus, every transition, except t_e and the dc-transitions, is enabled at m_1 in N_1 if and only if is enabled at m_2 in N_2. Transition t_e is never enabled, because σ is a correct run, and C does not halt by assumption (i.e., place s_n is never marked). We distinguish two cases:

1. The firing of any transition besides t_i^Z, t_i^C, and $t_i^{C'}$ (for $i \in BS(C)$) at m_1 in N_1 can be simulated by the firing of the same transition at m_2 in N_2, and vice versa. The respective firings lead again to a marking pair in M.
2. If cheating is possible in N_1 at m_1 and N_1 fires t_i^Z, t_i^C, or $t_i^{C'}$ with $i \in BS(C)$ when the respective place c_j is not empty, then one transition out of the set $\{t_i^Z, t_i^C, t_i^{C'}\}$ can fire in N_2 such that both nets have the same marking m (and thus $(m, m) \in D$) afterward. In detail: If $m_1 \xrightarrow{t_i^Z} m$ in N_1, then $m_2 \xrightarrow{t_i^{C'}} m$ in N_2; if $m_1 \xrightarrow{t_i^C} m$ in N_1, then $m_2 \xrightarrow{t_i^Z} m$ in N_2. The same argument applies if cheating is possible in N_2: If $m_2 \xrightarrow{t_i^Z} m$ in N_2, then $m_1 \xrightarrow{t_i^C} m$ in N_1; if $m_2 \xrightarrow{t_i^{C'}} m$ in N_2, then $m_1 \xrightarrow{t_i^Z} m$ in N_1.

If N_1 and N_2 have the same marking (i.e., we have a pair in D), then each can simulate the other by firing the same transition, remaining in D. Thus, $D \uplus M$ is a bisimulation.

(2) implies (3): trivial

(3) implies (1): By contraposition, assume C halts for inputs x_1 and x_2. Then, we construct a run $m_{N_1} \xrightarrow{\sigma} m$ in N_1 such that σ simulates C correctly (i.e., without cheating) and $m(s_n) = 1$ (i.e., C reaches the $HALT$ command): For each command CMD_i that C performs, we add three transitions to σ. If $i \notin BS(C)$, we add $v_i t_i v_i'$ to σ. If $i \in BS(C)$, we add $z_i t_i^Z z_i'$ (if the respective counter is zero) or $n_i t_i^N n_i'$ (otherwise) to σ. Now the trace w corresponding to the run $\sigma t_e f$ is a stop-trace of N_1, i.e., $w \in \mathit{stop}(N_1)$.

To perform the same trace in N_2, there is no choice but to perform the same run σ (except for possibly firing t_p or t_p' in-between): For instance, to perform action v_i one has to fire transition v_i, and to perform action v_i' then one has to fire transitions $t_i v_i'$. Observe that one cannot fire $t_i^C z_i'$ or $t_i'^C z_i'$ to perform action z_i' because the firing of t_i^Z is correct at this stage and, thus, the respective counter (and the corresponding place) is empty. However, after σ the transition t_e is not enabled in N_2, because p is not marked. Thus, $w \notin L(N_2)$, which implies $w \notin \mathit{stop}(N_2)$. □

With Lemma 14, we reduce *df*- and *fdf*-accordance to the halting problem of a 2-counter machine.

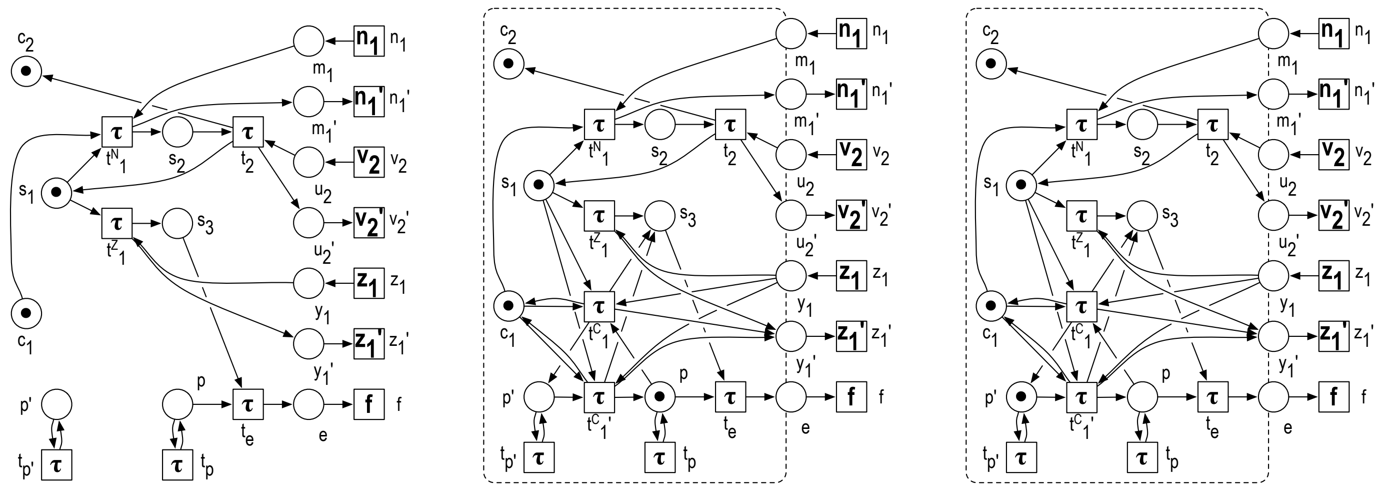
Theorem 15 (undecidability of *df*- and *fdf*-accordance)**.** *For two interface-equivalent open nets Impl and Spec, df-accordance and fdf-accordance are undecidable.*

Proof. Let C be a 2-counter machine with input values x_1 and x_2. We construct two interface-equivalent open nets $\mathit{open}(N_1)$ and $\mathit{open}(N_2)$ from the labeled nets N_1 and N_2 from Lemma 14 by removing all transitions t that are not τ-labeled, and interpreting t's preset (postset) as output (input) place. Figures 3b and 3c illustrate $\mathit{open}(N_1)$ and $\mathit{open}(N_2)$ for ADD, if we ignore all transitions outside the dashed frame. Clearly, $\mathit{stop}(\mathit{open}(N_1)) = \mathit{stop}(N_1)$ and $\mathit{stop}(\mathit{open}(N_2)) = \mathit{stop}(N_2)$. Now assume that *df*-accordance is decidable. Then $\mathit{open}(N_1)$ *df*-accords with $\mathit{open}(N_2)$ iff $\mathit{stop}(\mathit{open}(N_1)) \subseteq \mathit{stop}(\mathit{open}(N_2))$ by Theorem 10 iff C does not halt for the given inputs x_1 and x_2 by Lemma 14. Thus, we can decide the halting problem for 2-counter machines, which is a contradiction to Theorem 13. Therefore, *df*-accordance is undecidable.

As *fdf*- and *df*-accordance coincide for open nets with an empty set of final markings, we conclude the undecidability of *fdf*-accordance from the undecidability of *df*-accordance. □

4. Undecidability of *r*- and *fr*-accordance

We prove that *r*- and *fr*-accordance and their coarsest precongruences $\sqsubseteq^c_{r,acc}$ and $\sqsubseteq^c_{fr,acc}$ are undecidable, thereby



(a) Step one and step two (b) N_1 (c) N_2

Figure 3: Auxiliary constructions and the labeled nets N_1 and N_2 (ignoring the dashed frame) for Lemma 14 and the counter machine *ADD*.

following the proof strategy from Sect. 3. As we use the trace-based characterization of *r*- and *fr*-accordance from [6], we redefine the *stopdead*-semantics from Sect. 3 for responsiveness.

Definition 16 (*stopdead*-semantics for responsiveness)**.** *Let N be a labeled net. A marking m of N is an* r-stop except for inputs *if there is no $o \in \Sigma_{out}$ such that $m \stackrel{o}{\Longrightarrow}$; marking m is* r-dead except for inputs *if additionally there exists no final marking m' of N with $m \stackrel{\varepsilon}{\Longrightarrow} m'$. The* responsive *stopdead*-semantics *of a N is defined by the sets of traces $rstop(N) = \{w \mid m_N \stackrel{w}{\Longrightarrow} m \wedge m$ is an r-stop except for inputs$\}$ and $rdead(N) = \{w \mid m_N \stackrel{w}{\Longrightarrow} m \wedge m$ is r-dead except for inputs$\}$.*

Theorem 17 (*r*- and *fr*-accordance characterization [6])**.** *For interface-equivalent open nets Impl and Spec, the following holds: (1) Impl r-accords with Spec iff $rstop(Impl) \subseteq rstop(Spec)$. (2) Impl fr-accords with Spec iff $rstop(Impl) \subseteq rstop(Spec)$ and $rdead(Impl) \subseteq rdead(Spec)$.*

Similarly to Sect. 3, we reduce *r*- and *fr*-accordance to the halting problem of a 2-counter machine.

Lemma 18. *Let C be a 2-counter machine and $x_1, x_2 \in \mathbb{N}$. We can construct two action-equivalent labeled nets N_1 and N_2 (as modifications of $net(C)$) such that the following conditions are equivalent:*

1. *C does not halt for the given inputs x_1 and x_2.*
2. *N_1 and N_2 are bisimilar.*
3. *$rstop(N_1) \subseteq rstop(N_2)$.*

Proof. We construct the two action-equivalent labeled nets N_1 and N_2 from C and the input values x_1 and x_2 in the same four steps as in the proof of Lemma 14, with only one modification of step two: We additionally add a place o, a transition t_o, and arcs (t_p, o),(t'_p, o), and (o, t_o). Transition t_o is labeled with the output action t_o.

As long as any of the places p, p', and o is marked, the corresponding marking is not an r-stop except for inputs: The transition t_o is labeled with an output action and may fire. Thus, the only way to reach an r-stop except for inputs is to empty the place o by firing t_o, and to empty the places p, p' by firing the transitions t_e and f. The rest of the proof is analogous to the proof of Lemma 14. □

We immediately conclude undecidability of *r*- and *fr*-accordance from Lemma 18 and Theorems 17 and 13 with an argument as in the proof of Theorem 15.

Theorem 19 (undecidability of *r* and *fr*-accordance)**.** *For two interface-equivalent open nets Impl and Spec, r-accordance and fr-accordance are undecidable.*

In the following, we show that also the coarsest precongruence contained in (f)-responsive accordance is undecidable. Here, it is essential that $\sqsubseteq^c_{r,acc}$ and $\sqsubseteq^c_{fr,acc}$ can be characterized using the impossible futures semantics $\mathcal{F}^+(N)$ [9, 12] and a modification $\mathcal{F}^+_{fin}(N)$ of it, as shown in [6]. With this, it is not difficult to prove the following lemma.[1]

Lemma 20. *For two action-equivalent labeled nets N_1 and N_2, the following holds: (1) If N_1 and N_2 are bisimilar, then $N_1 \sqsubseteq^c_{r,acc} N_2$. (2) $N_1 \sqsubseteq^c_{r,acc} N_2$ implies $L(N_1) \subseteq L(N_2)$.*

With the construction from Lemma 18, we show the undecidability of the coarsest precongruence contained in each preorder.

[1]For bisimilar nets, even the $\mathcal{F}^+$-semantics coincide. If $N_1 \sqsubseteq^c_{r,acc} N_2$ and $w \in L(N_1)$ then $(w, \emptyset) \in \mathcal{F}^+(N_1)$, implying $(w, \emptyset) \in \mathcal{F}^+(N_2)$ and thus $w \in L(N_2)$.

Lemma 21. *Let C be a 2-counter machine and $x_1, x_2 \in \mathbb{N}$. We can construct two action-equivalent labeled nets N_1 and N_2 (without final markings) such that the following conditions are equivalent:*

1. *C does not halt for the given inputs x_1 and x_2.*
2. *$N_1 \sqsubseteq^c_{r,acc} N_2$.*

Proof. We construct the labeled nets N_1 and N_2 as in the proof of Lemma 18. (1) implies (2) because N_1 and N_2 are bisimilar by Lemma 18, which implies $N_1 \sqsubseteq^c_{r,acc} N_2$ by Lemma 20(1). (2) implies (1) because if C halts for the inputs x_1 and x_2, then $L(N_1) \not\subseteq L(N_2)$ as shown in the proof of Lemma 18 and 14). Thus, $N_1 \not\sqsubseteq^c_{r,acc} N_2$ by Lemma 20(2). □

One can observe that, for open nets without final markings, $\sqsubseteq^c_{r,acc}$ and $\sqsubseteq^c_{fr,acc}$ coincide.[2] With this, we immediately conclude undecidability of $\sqsubseteq^c_{r,acc}$ and $\sqsubseteq^c_{fr,acc}$ from Lemma 21 and Theorem 13 with an argument as in the proof of Theorem 15.

Theorem 22 (undecidability of $\sqsubseteq^c_{r,acc}$ and $\sqsubseteq^c_{fr,acc}$)**.** *For two interface-equivalent open nets Impl and Spec, the precongruences $\sqsubseteq^c_{r,acc}$ and $\sqsubseteq^c_{fr,acc}$ are undecidable.*

5. Undecidability of *wt*-accordance

Finally, we reduce to the decision of *wt*-accordance the question whether an open net has at least one *wt*-controller—that is, *wt*-controllability. As the latter is undecidable [13], *wt*-accordance is undecidable, too.

Theorem 23. *For two interface-equivalent open nets Impl and Spec, wt-accordance is undecidable.*

Proof. We reduce *wt*-controllability to *wt*-accordance. Given an open net N, we can construct an interface-equivalent open net C that is not *wt*-controllable (by putting $\Omega_C = \emptyset$). First, if C *wt*-accords with N, then every *wt*-controller of N is a *wt*-controller of C and, thus, N is not *wt*-controllable. Second, if C does not *wt*-accord with N, then N has at least one *wt*-controller (that is not a *wt*-controller of C) and, thus, N is *wt*-controllable. Hence, N is *wt*-controllable iff C does not *wt*-accord with it. □

Bravetti and Zavattaro [14] define the subcontract preorder which preserves weak termination. The model in [14] is a modified version of Milner's CCS [8] with one unbounded but ordered message queue. In contrast, in our Petri net model, each interface place models an unbounded unordered message queue. Therefore, Theorem 23 does not imply that the subcontract preorder in [14] is undecidable, but we suspect that it is.

[2] Essentially, for such an open net N, one can "embed" $\mathcal{F}^+(N)$ in $\mathcal{F}^+_{fin}(N)$ by adding $\emptyset$ as a third component to each element of $\mathcal{F}^+(N)$. Furthermore, for any set Y, $(w, X, Y) \in \mathcal{F}^+_{fin}(N)$ iff $(w, X, \emptyset) \in \mathcal{F}^+_{fin}(N)$.

6. Related Work and Conclusion

We showed undecidability of accordance for five behavioral properties: deadlock freedom [3] and responsiveness [4]—both with and without final markings—and weak termination [5].

Our proofs mostly work by reduction from the halting problem of 2-counter machines using a variation of the "Jančar-Pattern" [11]. Counter machines and their halting problem were introduced in [10]. The halting problem for counter machines can be used very naturally to show the undecidability of other problems related to Petri nets, such as bisimilarity and language inclusion [11, 15].

The controllability problem is decidable for deadlock freedom and responsiveness: There always exists a trivial controller with an internal loop (deadlock freedom) or a loop in which messages are sent without waiting for an answer (responsiveness). As the corresponding accordance preorders are undecidable, the accordance is a more difficult problem than controllability.

Future work is to investigate accordance for weak termination and bounded communication.

References

[1] M. P. Papazoglou, Web Services: Principles and Technology, Pearson, 2007.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (2002) 393 – 422.

[3] C. Stahl, P. Massuthe, J. Bretschneider, Deciding substitutability of services with operating guidelines, in: ToPNoC II, LNCS 5460, Springer, 2009, pp. 172–191.

[4] K. Wolf, Does my service have partners?, in: ToPNoC II, LNCS 5460, Springer, 2009, pp. 152–171.

[5] A. J. Mooij, C. Stahl, M. Voorhoeve, Relating fair testing and accordance for service replaceability, J. Log. Algebr. Program. 79 (2010) 233–244.

[6] W. Vogler, C. Stahl, R. Müller, A trace-based semantics for responsiveness, in: ACSD 2012, IEEE, 2012, pp. 42–51.

[7] C. Stahl, W. Vogler, A trace-based service semantics guaranteeing deadlock freedom, Acta Inf. 49 (2012) 69–103.

[8] R. Milner, Communication and Concurrency, Prentice-Hall, Inc., 1989.

[9] W. Vogler, Modular Construction and Partial Order Semantics of Petri Nets, volume 625 of *LNCS*, Springer, 1992.

[10] M. Minsky, Computation: Finite and infinite machines (1967).

[11] P. Jančar, Undecidability of bisimilarity for Petri nets and some related problems, Theoretical Computer Science 148 (1995) 281 – 301.

[12] A. Rensink, W. Vogler, Fair testing, Inf. Comput. 205 (2007) 125–198.

[13] P. Massuthe, A. Serebrenik, N. Sidorova, K. Wolf, Can I find a partner? Undecidablity of partner existence for open nets, Inf. Process. Lett. 108 (2008) 374–378.

[14] M. Bravetti, G. Zavattaro, Contract compliance and choreography conformance in the presence of message queues, in: WS-FM 2008, volume 5387 of *LNCS*, Springer, 2009, pp. 37–54.

[15] J. Esparza, Decidability and complexity of Petri net problems–an introduction, in: Lectures on Petri Nets I: Basic Models, Springer, 1998, pp. 374–428.