Can I find a partner? Undecidability of partner existence for open nets

Peter Massuthe, Alexander Serebrenik, Natalia Sidorova, Karsten Wolf

Humboldt-Universität zu Berlin, Institut für Informatik, Unter den Linden 6, 10099 Berlin, Germany
Eindhoven University of Technology, Department of Mathematics and Computer Science, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
Universität Rostock, Institut für Informatik, 18051 Rostock, Germany

Article history:
Received 22 January 2008
Available online 16 July 2008
Communicated by S.E. Hambrusch

Keywords:
Petri net
Open net
Operability
Web service
Theory of computation

1. Introduction

Service oriented computing (SOC) is about to become an established paradigm for the design of interorganizational workflows and for the programming-in-the-large. In practice, languages like WS-BPEL [2] have been developed and are increasingly used. These developments are accompanied by considerable efforts to support SOC with formal models.

One purpose of a formal model for a web service is the capability of performing formal analysis. In this article, we study the analysis problem of operability of open nets. Open nets [8] are a special subclass of classical Petri net models that explicitly model communication with the environment. The control flow of practically relevant languages such as WS-BPEL can be translated into open nets [7]. These translations by construction lead to bounded open nets.

An open net is called operable if there exists an environment such that the cooperation of the net and the environment leads to the desired final state. For bounded nets, in our previous work [8,11] we have proposed decision procedures for operability (also called controllability). In this paper we show that operability of open nets is, in general, undecidable, i.e., the restriction to bounded nets is essential for decidability. To this end we use a reduction from the language inclusion problem for Petri nets, known to be undecidable [5].

Related work. The notion of operability is related to two well-known behavioral notions. First of all, operability has been inspired by the well-studied notion of soundness for workflow nets, Petri nets used to model workflows [1]. The adaptation of soundness to the web service case has been discussed in [6,8,9,11]. Moreover, the studied problem is in fact a control theoretic problem where the given open net assumes the role of the plant while the partner serves as controller. Unfortunately, classical control theory [3,10] seems not to provide a result that is immediately applicable to our specific setting.

In [4] it is shown that model checking an LTL formula is undecidable for a service composition in a formal model comparable to open nets. This result is weaker than ours, i.e., we show that even a subclass of this problem is undecidable. Furthermore, [4] presents sufficient conditions under which the problem can be reduced to a finite state.
problem known to be decidable. The reduction is comparable to our restrictions of bounded communication [8].

2. Basic definitions

In this paper we consider a subclass of Petri nets, called open nets. We recall basic notions related to Petri nets and then introduce the subclass of interest.

2.1. Petri nets

\[ \mathbb{N} \] denotes the set of natural numbers.

Let \( P \) be a set. A bag \( m \) over \( P \) is a mapping \( m : P \rightarrow \mathbb{N} \). The set of all bags over \( P \) is denoted by \( \mathbb{N}^P \). We use \( + \) and \( - \) for the sum and the difference of two bags and \( =, <, >, \leq \) and \( \geq \) for the comparison of bags, which are defined in the standard way. We write, e.g., \( m = 2[p] + [q] \) for a bag \( m \) with \( m(p) = 2 \), \( m(q) = 1 \), and \( m(x) = 0 \) for all \( x \notin \{p, q\} \).

For finite sequences of elements over a set \( P \) we use the following notation: The empty sequence is denoted with \( \varepsilon \); a nonempty sequence can be given by listing its elements.

**Definition 1 (Petri net).** A Petri net \( N \) over a fixed set of labels \( Act \) is a tuple \( \langle P, T, F, \Lambda \rangle \), where:

1. \( P \) and \( T \) are two disjoint nonempty finite sets of places and transitions, respectively;
2. \( F : (P \times T) \cup (T \times P) \rightarrow \mathbb{N} \) is a flow relation mapping pairs of places and transitions to the naturals;
3. \( \Lambda : T \rightarrow Act \cup \{\varepsilon\} \) is a labeling function that maps transitions of \( T \) to action labels from \( Act \) or to the special label \( \varepsilon \) denoting a silent action.

Given a transition \( t \in T \), the preset \( \bullet t \) and the postset \( t \bullet \) of \( t \) are the bags of places where every \( p \in P \) occurs \( F(p, t) \) times in \( \bullet t \) and \( F(t, p) \) times in \( t \bullet \). Analogously we write \( P^t \) for the pre- and the postset of the place \( P \). We also write \( \bullet \sigma(t) \) and \( (t)P^t \) to explicitly denote the Petri net.

A marking \( m \) of \( N \) is a bag over \( P \); markings are states of a net. A pair \( (N, m) \) is called a marked Petri net. A transition \( t \in T \) is enabled in marking \( m \) if and only if \( t \leq m \). An enabled transition \( t \) may fire. This results in a new marking \( m' \) defined by \( m' = m - t + t \bullet \). In this case we also write \( m \xrightarrow{\sigma(t)} m' \). For a sequence of action names \( \sigma = a_1 \ldots a_{n-1} \) we write \( m \xrightarrow{\sigma} m' \) when \( m = m_1 \xrightarrow{a_1} m_2 \xrightarrow{a_2} \ldots \xrightarrow{a_{n-1}} m_n = m' \).

We furthermore write \( m \rightarrow m \) for any marking \( m \). We say that \( m \) is reachable from \( m \) if and only if there exists \( \sigma \in Act^* \) such that \( m \xrightarrow{\sigma} m' \). We denote the set of all markings reachable in net \( m \) from marking \( m \) as \( \mathcal{R}_N(m) \).

We further write \( m \Rightarrow m' \) when \( m = m_1 \Rightarrow m_2 \Rightarrow \ldots \Rightarrow m' \). We also write \( m \mathcal{L} \Rightarrow m' \) if \( m \Rightarrow m_1 \Rightarrow m_2 \Rightarrow m' \). The weak language \( \mathcal{L}^* (N, m_0) \) of a marked Petri net \( (N, m_0) \) is defined as \( \{ \sigma \in Act^* \mid \exists m : m_0 \mathcal{L} \Rightarrow m \} \). The strong language \( \mathcal{L} (N, m_0) \) of a marked \( \tau \)-free Petri net \( (N, m_0) \) is defined as \( \{ \sigma \in Act^* \mid \exists m : m_0 \xrightarrow{\sigma} m \} \).

2.2. Open nets

Open nets are a special class of Petri nets intended for modelling asynchronously communicating components. Interface ports are modelled as specially indicated places. Open nets can then be connected to each other by merging (some of) their interface places.

**Definition 2 (Open net).** An open net \( N \) is a tuple \( \langle P, T, F, \Lambda, P_i, P_o \rangle \), where:

1. \( \langle P \cup P_i \cup P_o, T, F, \Lambda \rangle \) is a Petri net;
2. \( P, P_i \) and \( P_o \) are disjoint, called internal, input and output places;
3. \( F(t, p_i) = F(p_o, t) = 0 \) for any \( p_i \in P_i \), \( p_o \in P_o \), and \( t \in T \);
4. for any \( p_i \in P_i \) there exists \( t \in T \) such that \( F(p_i, t) > 0 \);
5. for any \( p_o \in P_o \) there exists \( t \in T \) such that \( F(t, p_o) > 0 \).

The input places \( P_i \) correspond to channels for incoming messages, the output places \( P_o \) to channels for the outgoing messages.

As an example, Fig. 1 shows two online shops modeled as open nets \( N_1 \) and \( N_2 \). A customer of the left-hand side shop may login first, followed by an order message. The customer then has the choice to confirm (finish) the order or to abort. In the first case, the shop sends the ordered goods to the customer (deliver) and expects the payment. The right-hand side shop introduces a check whether the ordered good is in stock (modeled by transition available) or not (capacity reached). In the first case, the customer has the choice to place more orders (continue) or to finish whereas in the second case, he may choose to get all available goods delivered (rest) or to abort completely. If at least one item was bought, the shop sends the ordered goods to the customer (deliver) and expects the payment.

**Definition 3 (Composition of open nets).** Let \( N = \langle P, T, F, \Lambda, P_i, P_o \rangle \) and \( N' = \langle P', T', F', \Lambda', P'_i, P'_o \rangle \) be two open nets with \( P \cap P' = \emptyset \), \( T \cap T' = \emptyset \), \( P_i \cap P'_i = \emptyset \) and \( P_o \cap P'_o = \emptyset \). The composition \( N \oplus N' \) is the open net \( \langle P \cup P', T \cup T', F \oplus, (P_i \cup P'_i), (P_o \cup P'_o) \setminus (P_i \cup P'_i) \rangle \), where \( F \oplus(x, y) = F(x, y) \) if \( (x, y) \in (P \times T) \cup (T \times P) \), \( F'(x, y) \) if \( (x, y) \in (P' \times T') \cup (T' \times P') \), and 0 otherwise.

For every open net we identify a unique initial marking \( m_0 \) of the internal places, and a nonempty and possibly infinite set \( \Omega \) of final markings of the internal places, corresponding to correct termination states of the open net. The interface places should be empty both at the beginning and at the end of the process. The main result of the paper holds also if one generalizes this emptiness requirement by demanding that the process preserves tokens at the interface places, i.e., \( m_0(p_i) = m(p_i) \) and \( m_0(p_o) = m(p_o) \) for any \( p_i \in P_i \), \( p_o \in P_o \) and \( m \in \Omega \). However, to keep the notions consistent with \([8,11]\) we opted for a more restricted version.
Given two open nets $N = \langle P, T, F, A, P_1, P_0 \rangle$ and $N' = \langle P', T', F', A', P'_1, P'_0 \rangle$ with initial markings $m_0$ and $m'_0$, respectively, the initial marking $m_0 \oplus m'_0$ of $N \oplus N'$ is defined as $(m_0 \oplus m'_0)(p) = m_0(p)$ if $p \in P$, $(m_0 \oplus m'_0)(p) = m'_0(p)$ if $p \in P'$ and $(m_0 \oplus m'_0)(p) = 0$ for $p \in (P_1 \cup P_0 \cup P'_1 \cup P'_0)$. The set $\Omega \oplus \Omega'$ of final markings is defined as the set of the markings that correspond to the compositions of all pairs of markings from $\Omega$ and $\Omega'$. Note that the composition of an arbitrary number of open nets is an open net again.

One clearly expects that an open net can operate properly at least in some environment. By proper operation we mean that the composition can still reach a final marking at any execution moment. We capture this requirement in the notion of operability (cf. [11]).

**Definition 4 (Operability).** An open net $N = \langle P, T, F, A, P_1, P_0 \rangle$ with a given initial marking $m_0$ and a given set of final markings $\Omega$ is called operable if there exists an open net $N' = \langle P', T', F', A', P'_0, P'_1 \rangle$ with an initial marking $m'_0$ and a set of final markings $\Omega'$ such that for any marking $m \in \mathcal{R}_{N \oplus N'}(m_0 \oplus m'_0)$ there exists a marking $m_f \in \Omega \oplus \Omega'$ such that $m_f \in \mathcal{R}_{N \oplus N'}(m)$. $N'$ as above is called a partner of $N$.

Reconsider the two online shops of Fig. 1. Let $\Omega_{N_1} = \Omega_{N_2} = \langle [p_1, p_2], [p_3] \rangle$ be the corresponding sets of final markings. It is easy to see that the open net $N_1$ is operable. The internal choice in $N_2$, however, is not communicated to the customer causing the net to be non-operable: after ordering the first item, the customer had to guess whether to send continue/finish or to send abort/rest. Whatever choice the partner makes, it might be the wrong one and the composed net is in a deadlock state and cannot reach a final marking.

### 3. Undecidability of operability

In this section we present the main result: undecidability of operability of open nets. We start by recalling the undecidability of Petri net language inclusion, i.e., whether or not $\mathcal{L}(N_1, m_1) \subseteq \mathcal{L}(N_2, m_2)$ and $\mathcal{L}^=(N_1, m_1) \subseteq \mathcal{L}^=(N_2, m_2)$.

**Theorem 5 (Theorem 10.2 from [5]).** Language inclusion for Petri nets is undecidable both for strong and weak languages.

In the following we prove undecidability of operability. We assume that it is decidable and show that this assumption implies that language inclusion for Petri nets should be decidable as well, contradicting Theorem 5.

**Theorem 6.** Operability is undecidable.

**Proof.** We prove undecidability by providing a construction for an open net $N = \langle P, T, F, A, P_1, P_0 \rangle$ based on given $\tau$-free Petri nets $N_1 = \langle P_1, T_1, F_1, A_1 \rangle$ and $N_2 = \langle P_2, T_2, F_2, A_2 \rangle$ with the initial markings $m_1$ and $m_2$, respectively, such that $N$ is operable if and only if the strong language of $(N_1, m_1)$ is not included in the strong language of $(N_2, m_2)$. In other words, a partner, if exists, should “provide” $N$ with a word in $\mathcal{L}(N_1, m_1)$ and not in $\mathcal{L}(N_2, m_2)$.

**Construction.** We build an auxiliary net $N'_1 = \langle P'_1, T'_1, F'_1, A'_1 \rangle$ by substituting every transition $t \in T_1$ by a sequence of the start-transition $t_i$ and the finish-transition $t_f$ connected via the place $\bar{p}_i$ such that $\mathcal{N}_1(t_i) = \mathcal{N}_1(t)$, $(t)f'_{N'_1} = (t)\mathcal{N}_1$, and $(t)s'_{N'_1} = s(f'_{N'_1}) = [\bar{p}_i]$. Furthermore, we introduce an additional place $s$ such that $s = \{t_f \mid t \in$
We further define $A'_1(t_s) = A_1(t)$, $A'_2(t_f) = \tau$ and $m'_1 = m_1 + [s]$. Clearly, $\Sigma(N_1, m_1) = \Sigma(N'_1, m'_1)$.

Now we add a discriminator construction (see Fig. 2) whose functioning will result either in a token on place $\text{ack}$ not belonging to any final marking, or in a token on place $\text{final}$ that belongs to every final marking of $N$. The token on trap is reached only if net $N_2$ has repeated all the steps of $N'_1$. Initially the discriminator contains a token on place $\text{ok}$ while the $\text{nok}$ place is empty, i.e. the firings of transitions of $N_2$ are not restricted, while the firings of transitions from $T_A$ are forbidden. In case $N_2$ is not able to repeat a step of $N'_1$, the only way to proceed is to move the token from $\text{ok}$ to $\text{nok}$, thus enabling the firings of transitions from $T_A$ and disabling the firing of transitions of $N_2$ forever. In this case we cannot reach the $\text{[trap]}$ marking anymore but can reach the $\text{[final]}$ marking. Note that the reject transition can also be taken when $N_2$ would be able to mimic the step of $N'_1$. For any reachable marking $m$ of $N$, $m(\text{ok}) + m(\text{nok}) + m(\text{final}) + m(\text{trap}) = 1$.

Finally, we extend our net with the interface places: for each transition $t$ in $T_1$ one input place $p_t$, a special input place $\text{ready}$ and a special output place $\text{ack}$. By placing tokens in $p_t$ a partner “agrees” to a firing of $t$ if it is enabled in $N_1$. By placing a token in $\text{ready}$ the partner indicates that the input is terminated and the discriminator may move a token to the $\text{final}$ or $\text{trap}$ places. The $\text{ack}$ place serves for sending a confirmation from $N'_1$ that a step has been taken.

The initial marking of $N$ is $m'_1 + m_2 + [ok]$. The set $\Omega$ of final markings is defined as the set of all markings of $N$ that contain a token on the $\text{final}$ place. Note that for every reachable marking $m$, $m(\text{ack}) + m(s) + m(\text{final}) + m(\text{trap}) + \sum_{t \in A'_1(T_1)} m(p_t) = 1$. That is, the place $s$ enforces the alternation of start- and finish-transitions in $N'_1$ with an intermediate response of $N_2$ until either $\text{final}$ or $\text{trap}$ is marked.

Correctness proof. ($\Rightarrow$) First we show that $\Sigma(N_1, m_1) \not\subseteq \Sigma(N_2, m_2)$ implies that $N$ is operable. Since $\Sigma(N_1, m_1) \not\subseteq \Sigma(N_2, m_2)$, there exists a firing sequence $\sigma = t_1t_2\ldots t_n \in T^*_1$ corresponding to a word $w \in \Sigma(N_1, m_1) \setminus \Sigma(N_2, m_2)$. Take the partner net $N^P = \langle P^P, T^P, F^P, \Lambda^P, P^1, P^0 \rangle$ (see Fig. 3) with the initial marking $[p_1]$ and the set of final markings $\{[q_{n+1}]\}$ where
- $P^P = \{p_1, \ldots, p_{n+1}\} \cup \{q_1, \ldots, q_{n+1}\}$,
- $T^P = \{u_1, \ldots, u_{n+1}\} \cup \{v_1, \ldots, v_n\}$,
• \( P^0 = \{ \text{ack} \} \), \( P^0 = \{ p_i \mid t_j \in \sigma \} \cup \{ \text{ready} \}. \)

• \( u_j = [p_j], \ j = 1, \ldots, n + 1; \ u_j^* = [q_j, p_i], \ j = 1, \ldots, n; \ n \)

• \( u_{n+1}^* = [q_{n+1}, \text{ready}] \).

• \( v_j = [q_j, \text{ack}], \ j = 1, \ldots, n; \ v_j^* = [p_{j+1}], \ j = 1, \ldots, n. \)

• \( \Lambda^p(x) = \tau \) for any \( x \in T^p \).

Since \( w \in \mathcal{L}(N_1, m_1) \setminus \mathcal{L}(N_2, m_2) \), it also holds that \( w \in \mathcal{L}^*(N'_1, m'_1) \setminus \mathcal{L}(N_2, m_2) \). Assume for the sake of contradiction that there exists an execution sequence in \((N \oplus N^p, (m'_1 + m_2 + [ok]) \oplus [p_1])\) resulting in \( \text{trap} \) being marked. Then, transition eq should have been fired, i.e., a token from \( \text{ready} \) has been consumed. Since, \( w \in \mathcal{L}^*(N'_1, m'_1) \setminus \mathcal{L}(N_2, m_2) \), there exists \( q_i \) such that \( N^p \) will get stuck on it due to missing acknowledgment from \( N \). Hence, no token can ever be placed in \( \text{ready} \), providing the desired contradiction.

Therefore, every execution sequence in \((N \oplus N^p, (m'_1 + m_2 + [ok]) \oplus [p_1])\) will contain the firing of the \( \text{reject} \) transition, i.e., will lead to \( \text{final} \) and not to \( \text{trap} \), implying that \( N \) is operable.

\( (\Rightarrow) \) Now suppose that \( \mathcal{L}(N_1, m_1) \subseteq \mathcal{L}(N_2, m_2) \). Since \( \mathcal{L}(N_1, m_1) = \mathcal{L}^*(N'_1, m'_1) \), \( N_2 \) can mimic any firing sequence of \( N'_1 \) and the discriminator can avoid the firing of the \( \text{reject} \) transition. Note that any final marking of \( N \) contains a token on the place \( \text{final} \), which can only be reached after the token on \( \text{ready} \) has appeared. However whenever a token on \( \text{ready} \) appears \( N \) is free to choose the firing of the transition \( eq \) leading to a non-final terminal marking. \( \Box \)

4. Conclusions

We proved undecidability of the operability problem for general open nets. We argued with a reduction from the language inclusion problem for Petri nets which is known to be undecidable.

Given the undecidability result stated in Theorem 6, to achieve decidability of operability one has to consider restricted classes of open nets. In [11], we proposed an algorithm for deciding operability in the case of acyclic open nets. [8] provides a decision procedure for open nets with three restrictions: First, it assumes an open net with finitely many inner markings (inner markings are abstract from tokens on interface places). Second, we only study deadlock freedom of the composed system which is weaker than operability. Third, existential quantification is restricted to those partners where communication does never put more then \( k \) tokens on an interface place, for some given \( k \). Despite the second restriction, we believe that operability as such is decidable under the first and third restriction.

As the future work we consider investigating decidability of operability for the following two classes of open nets. First of all, we plan to consider open nets with finitely many inner markings and partners that never put more then a certain number of tokens on an interface place. Unlike the class studied in [8] the value of \( k \) is not given. Second, we would like to investigate open nets with finitely many inner markings and partners that are not restricted in the number of tokens they put on interface places.

References


\footnote{Note that if \( t_i \) coincides with \( t_j \) for some \( 1 \leq i < j \leq n \), the corresponding output places \( p_i \) and \( p_j \) coincide as well.}