Why does my service have no partners?

Niels Lohmann

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
niels.lohmann@uni-rostock.de

Abstract. Controllability is a fundamental correctness criterion for interacting service models. A service model is controllable if there exists a partner service such that their composition is free of deadlocks and livelocks. Whereas controllability can be automatically decided, the existing decision algorithm gives no information about the reasons of why a service model is uncontrollable. This paper introduces a diagnosis framework to find these reasons which can help to fix uncontrollable service models.

Key words: Controllability, diagnosis, partner synthesis, verification

1 Introduction

In the paradigm of service-oriented computing [1, 2], a *service* is an component that offers a functionality over a well-defined interface and is discoverable and accessible via a unique identifier. By composing several services, complex tasks (e. g., inter-organizational business processes) can be realized. Thereby, the correct interplay of distributed services is crucial to achieve a common goal.

A fundamental correctness criterion for a service model is *controllability* [3]: A service S is controllable if there exists a partner service S' such that their composition is free of deadlocks and livelocks; that is, a desired final state is always reachable. The algorithm proposed to decide controllability [3] is constructive: If a partner service for S exists, it can be synthesized and serves as a witness for the controllability of S. If, however, S is uncontrollable, such a partner service does not exist and the decision algorithm returns no service. Obviously, this does not give any information about the reasons of *why* S is uncontrollable. Nevertheless, diagnosis information are important to support the modeller to correct the ill-designed service (see [4] for first results on service correction).

Controllability of a service model has a close relationship to soundness in the area of workflow models [5]. Intuitively, for a controllable service exists another service that their composition is a sound workflow. However, existing diagnosis techniques for unsound workflow models [6] are not applicable to diagnose uncontrollability, because the service's environment has to be taken into account.

In this paper, we study the reasons of uncontrollability and define a diagnosis algorithm that calculates a counterexample why a service model is uncontrollable. This counterexample can be used to repair the service model towards controllability. The paper employs *open nets* [7], a class of Petri nets [8], as formal model.

Niels Lohmann. **Why does my service have no partners?** In Roberto Bruni and Karsten Wolf, editors, *Web Services and Formal Methods, Fifth International Workshop, WS-FM 2008, Milan, Italy, September 4–5, 2008, Proceedings*, Lecture Notes in Computer Science. Springer-Verlag, September 2008.

With translations [9, 10] from and to WS-BPEL [11], the results can be directly applied to industrial service specification languages.

The rest of this paper is organized as follows. The next section recalls the required definitions on open nets, controllability and its decision algorithm. These definitions are crucial to understand the diagnosis framework and therefore are quite detailed. Reasons that can make a net uncontrollable are described in Sect. 3. The main contribution of this paper is a diagnosis algorithm for uncontrollable nets which is described in Sect. 4. Section 5 concludes the paper and gives directions for future research.

2 Formal Models for Services

2.1 Open Nets

Open nets [7] extend classical Petri nets [8] with an interface for asynchronous message exchange. We assume the standard firing rule and denote the set of all markings of a net N that are reachable from m with $\mathcal{R}_N(m) = \{m' \mid m \xrightarrow{*}_N m'\}$. $m \xrightarrow{t}_N$ denotes that in N, the marking m enables the transition t, and $m \not\rightarrow_N$ denotes that m does not activate any transition (i. e., m is a deadlock).

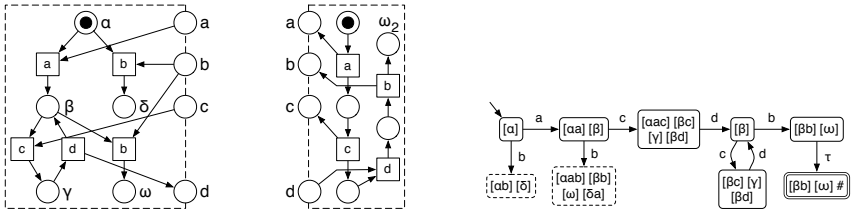
Definition 1 (Open Net, inner structure). *An* open net *is a Petri net $N = [P, T, F, m_0]$, together with an* interface $I = P_{in} \cup P_{out}$ *such that $P_{in}, P_{out} \subseteq P$, $P_{in} \cap P_{out} = \emptyset$, and $p \in P_{in}$ (resp. $p \in P_{out}$) implies ${}^\bullet p = \emptyset$ (resp. $p^\bullet = \emptyset$); and a set Ω of* final markings. *P_{in} (resp. P_{out}) is called the set of* input *(resp.* output*)* places. *We further require that $m(p) = 0$ for all $p \in I$ and $m \in \Omega \cup \{m_0\}$, and that no marking $m_f \in \Omega$ enables a transition. N is called* closed *iff $|I| = \emptyset$. N is called* normal *iff every transition of N is connected to at most one interface place. For a normal net N, define the mapping $\ell : T \to I \cup \{\tau\}$ such that $\ell(t)$ is the interface place connected to t if one exists, and $\ell(t) = \tau$ otherwise. The* inner structure *of N is the closed net $Inner(N) = [P \setminus I, T, F \setminus ((I \times T) \cup (T \times I)), m_{0|P\setminus I}, \emptyset, \emptyset, \Omega_{|P\setminus I}]$.*

The interface places are unidirectional: messages sent to N or received from N cannot be "unsent" or "unreceived", respectively. The inner structure of N models the control flow of N and can be compared to a classical workflow net [5].[1] In the rest of this paper, we only consider normal open nets. This restriction is rather technical, because every open net can be transformed [12] into a normal open net, and nets translated [9] from WS-BPEL processes are normal by construction.

Figure 1(a) depicts an open net N_1. We follow the standard graphical notation for Petri nets. The initial marking $m_0 = [\alpha]$ is depicted by a token on place α. We further assume $\Omega = \{[\omega]\}$ to be the set of final markings. Interface places ($I = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$) are located on the dashed frame, and the transition label $\ell(t)$ is written inside the respective transition t.

An open net N is *bounded* iff $\mathcal{R}_{Inner(N)}(m_{0|Inner(N)})$ is finite. Throughout this paper, we only consider bounded open nets, because for unbounded nets,

[1] Open nets do not share the structural constraints of workflow nets and allow arbitrary initial and final markings.



(a) open net N_1 (b) open net N_2 (c) overapproximated partner for N_1

Fig. 1. Two open nets and an overapproximated partner.

controllability is undecidable [13]. Boundedness can be verified using standard state space verification techniques [14]. Again, open nets translated from WS-BPEL processes are bounded by construction. See [9] for details.

Open nets can be composed by connecting the interfaces appropriately:

Definition 2 (Composition). *Two open nets N_1 and N_2 are* composable *iff $P_{in1} = P_{out2}$, $P_{out1} = P_{in2}$, $P_1 \cap P_2 = I_1 \cup I_2 \neq \emptyset$, and $T_1 \cap T_2 = \emptyset$. The* composition *of two composable open nets N_1 and N_2, denoted $N_1 \oplus N_2$, is the closed net with the following constituents: $P = P_1 \cup P_2$, $T = T_1 \cup T_2$, $F = F_1 \cup F_2$, $m_0 = m_{01} \oplus m_{02}$, $P_{in} = P_{out} = \emptyset$, and $\Omega = \{m_{f1} \oplus m_{f2} \mid m_{f1} \in \Omega_1, m_{f2} \in \Omega_2\}$. The composition of markings is defined as $m_1 \oplus m_2(p) = m_1(p)$ if $p \in P_1$, and $m_1 \oplus m_2(p) = m_2(p)$, otherwise.*

Definition 3 (Trapped marking). *Let N be a normal open net. A strongly connected component (SCC) is a maximal set of markings $\mathcal{M}$ of N such that $m, m' \in \mathcal{M}$ implies $m \xrightarrow{*}_N m'$. A terminal SCC (TSCC) is a SCC from which no other SCC is reachable. A marking m of N is a* trapped marking *if $m \in \mathcal{M}$ for a TSCC $\mathcal{M}$ and $\mathcal{M} \cap \Omega_N = \emptyset$. If additionally $m_{|Inner(N)} \xrightarrow{t}_{Inner(N)}$, $m \not\xrightarrow{t}_N$, and $\ell(t) \cap P_{in} \neq \emptyset$ then m is a* resolvable trapped marking *and $\ell(t)$* resolves *m. If no such t exists and $m_{|Inner(N)} \in \Omega_{Inner(N)}$, m is a covered final marking. If no such t exists and $m_{|Inner(N)} \notin \Omega_{Inner(N)}$, m is an* unresolvable trapped marking.

Resolvable trapped markings model situations the net can only leave with communication (i. e., message receipt). In contrast, unresolvable trapped markings can never be left. Covered final markings model situations in which the inner of an open net reached a final marking, but messages are still pending on some input places. Note that by definition of Ω, the TSCC of a covered final marking is a singleton; that is, a covered final marking is a deadlock.

Definition 4 (Weak termination, k-controllability, message bound). *A closed net N* weakly terminates *iff, for every marking $m \in \mathcal{R}_N(m_0)$, a final marking $m_f \in \Omega$ is reachable. A normal, bounded open net N is k-*controllable *for $k \in \mathbb{N}^+$ (called* message bound*) iff there exists a normal, bounded open net M such that N and M are composable, $N \oplus M$ weakly terminates, and for all $m \in \mathcal{R}_{N \oplus M}(m_{0N} \oplus m_{0M})$ holds that $m(p) \leq k$ for all $p \in I$.*

It is easy to check that the open nets N_1 and N_2 (see Fig. 1(a) and Fig. 1(b)) are bounded and composable. With $\Omega_{N_2} = \{[\omega_2]\}$, we have $\Omega_{N_1 \oplus N_2} = \{[\omega, \omega_2]\}$.

The composition weakly terminates, and for all reachable markings m, it holds $m(p) \leq 1$ for all $p \in I_{N_1}$. Thus, the open net N_1 is 1-controllable.

2.2 Deciding Controllability

For the rest of this section, let N be a normal, bounded open net, and let $k \in \mathbb{N}^+$. In the following, we sketch the decision algorithm for controllability presented in [15] which extends the one of [3]. The next definition synthesizes a transition system as first overapproximation of a partner of N. Thereby, each state consists of a set of markings N can reach after each communication step. The element "#" is a special symbol to denote final states.

Definition 5 (Partner overapproximation $\mathcal{TS}_0$). *For a set of markings $\mathcal{M}$ of N, define closure$(\mathcal{M}) = \bigcup_{m \in \mathcal{M}} \mathcal{R}_N(m)$. Let $\mathcal{TS}_0 = [Q, E, q_0, Q_F]$ be a transition system (consisting of a set Q of states, a set $E \subseteq Q \times (I \cup \{\tau\}) \times Q$ of labeled edges, an initial state $q_0 \in Q$, and a set $Q_F \subseteq Q$ of final states) inductively defined as follows:*

1. *$q_0 =$ closure$(\{m_0\})$; $q_0 \in Q$;*
2. *If $q \in Q$, then*
   (a) *if $\# \notin q$ and $q \cap \Omega \neq \emptyset$, then $q' = q \cup \{\#\} \in Q$, $q' \in Q_F$, and $[q, \tau, q'] \in E$;*
   (b) *if $m^* \in q$ is a resolvable trapped marking that can be resolved by x, $\# \notin q$, then $q' =$ closure$(\{m + [x] \mid m \in q\}) \in Q$ and $[q, x, q'] \in E$;*
   (c) *if $m^* \in q$ with $m^*(x) > 0$ for an $x \in P_{out}$, then $q' = \{m - [x] \mid m \in q, m(x) > 0\} \setminus \{\#\} \in Q$ and $[q, x, q'] \in E$.*

Thereby, $[x]$ denotes the marking with $[x](x) = 1$ and $[x](y) = 0$ for $y \neq x$. The operations $+$ and $-$ on markings are defined pointwise.

A transition system $\mathcal{TS}$ can be composed with an open net N in a canonic way (see [12] for details). In particular, we identify states of $N \oplus \mathcal{TS}$ with $[m, q]$ where m is a marking of N and $q \in Q$ is a state of $\mathcal{TS}$. Thereby, a path σ in the transition system $\mathcal{TS}$ can be extended to the composition $N \oplus \mathcal{TS}$. As each state of $\mathcal{TS}$ can contain multiple markings, this resulting path may not be unique.

The following definition removes all states from the overapproximation $\mathcal{TS}_0$ that jeopardize weak termination of the composition $N \oplus \mathcal{TS}_0$.

Definition 6 (Interaction graph $\mathcal{TS}^*$). *Let $\mathcal{TS}_1$ be the graph that is obtained from $\mathcal{TS}_0$ by removing all states q that contain a marking m with $m(p) > k$ for a $p \in I_N$. Given $\mathcal{TS}_i$ $(i > 0)$, the graph $\mathcal{TS}_{i+1}$ is obtained by removing each state q from $\mathcal{TS}_i$ that contains a marking m where no state $[m_f, q_f]$ is reachable in $N \oplus \mathcal{TS}_i$ from $[m, q]$ where $q_f \in Q_{F_{\mathcal{TS}_i}}$ and $m_f \in \Omega$.*

Thereby, removal of a state includes removal of its adjacent arcs and all states that become unreachable from q_0. Let $\mathcal{TS}^*$ be $\mathcal{TS}_j$ for the smallest j with $\mathcal{TS}_j = \mathcal{TS}_{j+1}$. $\mathcal{TS}^*$ is the interaction graph *of N.*

Theorem 1. *N is k-controllable iff $Q_{\mathcal{TS}^*} \neq \emptyset$.*

A proof to a similar theorem can be found in [15]. Intuitively, each state that is removed in Def. 6 cannot be part of a controlling partner. Thereby, the step from $\mathcal{TS}_0$ to $\mathcal{TS}_1$ assert that the message bound is respected and all future state removals assert weak termination. Note that τ-edges are a necessary technicality to assure livelock freedom. The interested reader is referred to [15].

Figure 1(c) depicts $\mathcal{TS}_0$ for the open net N_1. This graph coincides with $\mathcal{TS}_1$, because no marking exceeds the message bound of $k = 1$. From $\mathcal{TS}_1$, the dashed states are removed, because they contain the unresolvable trapped markings $[\delta]$ and $[\delta\mathsf{a}]$, respectively, from which the final marking is unreachable. The resulting graph without dashed states is the interaction graph $\mathcal{TS}^*$ of N_1. From this graph, we can conclude that a partner may only send a b-message after having received a d-message, because only then the partner can be sure that N_1 has left its initial marking. The open net N_2 (cf. Fig. 1(b)) models one possible partner of N_1.

3 Reasons for Uncontrollability

Obviously, the empty interaction graph (i. e., a graph with an empty set of states) calculated in case of uncontrollability gives no information why N has no partners and how N can be fixed. Before we examine uncontrollable nets, we study a related diagnosis approach.

3.1 Relationship to Soundness

For a controllable open net N exists an open net M such that $N \oplus M$ weakly terminates. Weak termination is closely related to soundness [5]. For soundness, an elaborate diagnosis algorithm exists [6] which exploits several properties of the soundness criterion to avoid a complex state space exploration where possible. For example, soundness can be expressed in terms of two simpler properties, namely liveness and boundedness. An unsound workflow net fails one of these tests. This result can be used to give detailed diagnosis information. In addition, several simple necessary or sufficient criteria for soundness can be checked prior to liveness and boundedness checks. For example, certain net classes such as free choice Petri nets [16] allow for efficient analysis algorithms.

However, this diagnosis approach cannot be adapted to diagnose the reasons of why an open net is uncontrollable. Firstly, a sound control flow is not related to controllability: the inner structure of the controllable net in Fig. 1(a) is not sound, and the uncontrollable net in Fig. 4(a) has a sound inner structure. Similarly, weaker criteria such as *relaxed soundness* or *non-controllable choice robustness* [17] are not applicable. The latter, for example, assumes that the environment (i. e., a partner) can completely observe the net's state, whereas the internal state of an open net can only be guessed from observations on the interface (a state of the interaction graph is a *set* of markings of the open net). Secondly, controllability is not a local, but a global criterion: controllability is now known to be decomposable. Therefore, only trivial necessary or sufficient criteria for (un)controllability exists. Thirdly, structural results like the invariant

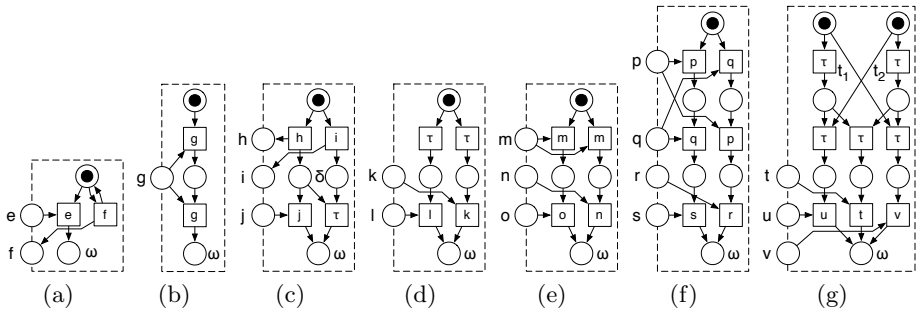


Fig. 2. Uncontrollable open nets ($k = 1$).

calculus [18] are not applicable, because an open net usually deadlocks when considered in isolation.

3.2 Classifying Harmful Markings

During the construction of the interaction graph of N, states that contain a marking m^* that is considered harmful are iteratively removed. The nature of m^* can be mapped back to the open net under consideration to understand the reasons that led to uncontrollability:

Inappropriate Message Bound. The message bound k influences the removal of states of $\mathcal{TS}_0$ when constructing $\mathcal{TS}_1$. This removal can make states containing a final marking unreachable. There are two situations to consider:

Unbounded communication. If a message channel is unbounded (e. g., due to a loop of the open net in which it sends messages without waiting for acknowledgements), then obviously no partner can exist such that the composition is k-bounded. Figure 2(a) shows an example where the output place f is unbounded.

Exceeded message bound. If a net is k-controllable for some bound $k \in \mathbb{N}^+$, it is also l-controllable for any $l > k$. However, the converse does not hold: Figure 2(b) shows a net that is 2-controllable, but not 1-controllable, because the receipt of the first g-message cannot be enforced before sending a second g-message. Thus, even if a message bound exists for a net, it might be considered k-uncontrollable if the message bound k chosen for analysis is too small.

Unreachable Final Marking. Definition 6 removes all states of the composition from which a final marking is unreachable. This can be due to a deadlock or a livelock in the composition. From the nature of a trapped marking, we can derive different diagnosis information. Hence, we classify different kinds of trapped markings of N that can be reached while communicating with a partner (cf. Def. 3).

Unresolvable trapped markings. If the inner of the net reaches a deadlock (called an *internal deadlock*), it cannot be left by communication with a partner. There are different reasons the control flow can contain deadlocks:

- *Design flaws.* The reason of an internal deadlock can be a classical design flaw as known from unsound workflow nets. For example, the deadlock $[\delta]$ of the open net in Fig. 2(c) is caused by mismatching splits and joins and cannot be avoided by communication, because the net internally decides whether to send an h-message or an i-message.
- *Service choreographies.* Internal deadlocks need not stem from modeling errors. One source of internal deadlocks can also be the composition of several services[2] in a service choreography. There, it is possible that the behavior of two participants A and B is mutually exclusive which leads to an internal deadlock. This deadlock, however, might be avoided by the message exchange with another participant C. Then C is a controlling partner for $A \oplus B$.
- *Behavioral constraints.* Another reason for internal deadlocks that are no design flaws can be behavioral constraints [19]. Such a constraint C influences the control flow and the final markings of an open net N such that some final markings of N are considered as internal deadlocks in the constrained net N_C.

Covered final markings. A covered final marking models a situation in which the control flow of N reached a final marking, but a message sent to N is still pending on a channel and will never be received. This might be negligible for generic acknowledgement messages, but in general, an unreceived message models an undesired situation (e. g., if the message contains private or payment information). Again, there are numerous reasons for this problem:

- *Hidden choices.* The net can make an internal decision (e. g., with WS-BPEL's `<if>` activity) that is not communicated to the partner (thus hidden), but requires different reactions of the partner depending on the outcome of the decision. Consider for example the net in Fig. 2(d) which either chooses the left or the right branch. Depending on this choice, a partner has to send a k-message or an l-message. The final marking is only reached, if the partner's "guess" was right. Otherwise, the "wrong" message keeps pending.
- *Conflicting receives.* If the net can reach a marking in which more than one transition is activated that can receive the same message (e. g., the initial marking of the net in Fig. 2(e)), these transitions are "conflicting receives". The decision which of these transitions fires can neither be influenced nor observed by a partner yielding a "hidden choice" situation. Note that conflicting receives are treated as runtime faults in execution languages like WS-BPEL. Like internal deadlocks, we do not want to forbid such situations in the first place, but check whether these problems are actually the reason why a service is uncontrollable.

[2] Composition as defined in Def. 2 is restricted to exactly two services, but can be canonically extended to allow for composition of an arbitrary number of open nets.

– *Pending messages.* Open nets model asynchronous message exchange: messages can keep pending on a channel, and overtake each other. Therefore, a partner has only limited control over the net, because after sending a message to N, a partner cannot be sure that N actually received that message. Again, this can result in a "hidden choice" situation, see the net in Fig. 2(f) for an example. For either branch, a p-message and a q-message message has to be sent eventually. After sending these messages, the environment has to guess whether to continue with sending an r-message or an s-message.
– *Confusion.* In a Petri net model, confusion [8] is a situation in which concurrency and conflicts coexist. Two concurrent transitions can influence each other without being in conflict; that is, whichever transition fires first can constrain the markings reachable by firing the other transition. Consider the net in Fig. 2(g) for an example: The transitions t_1 and t_2 are not in conflict, but whichever transition fires first, confines the choices reachable by firing the other transition. For example, firing first t_1 makes the firing of the transition receiving the v-message impossible. Now a partner has to guess whether to send a t-message or a u-message.

Livelocks. A livelock in the communication between N and a partner M is a situation in which in $N \oplus M$ transitions are fired continuously, but a final marking can never be reached. While this might be acceptable for technical systems where the concept of a "final state" is not applicable or irrelevant, services that implement business processes are usually required to reach a state in which a business instance is properly finished.

Note that the characterization of harmful markings as well as the definition of soundness is a behavioral and not a structural criterion. The nets depicted in Fig. 2 cannot be used as "anti-patters". If an open net contains one of these nets as a subnet, it might still be controllable. For example, prior communication might exclude the harmful subnet (e. g., using a deferred choice) or a concurrent or subsequent subnet "fixes" the communication of the net (e. g., by receiving pending message).

In the following section, we will present an algorithm to diagnose the problems described above except the problem of "unbounded communication". In the rest of this paper, we do not consider open nets without a fixed message bound. We assume that the message bound k is known prior to the controllability analysis. The value of k may be chosen by considerations on the physical message channels, by a static analysis that delivers a "sufficiently high" value, etc.

4 Diagnosing Reasons for Uncontrollability

4.1 Counterexamples for Controllability

In the area of computer-aided verification, model checking techniques [14] usually provide a *counterexample* if a model does not meet a specification. This counterexample is a useful artifact (e. g., a deadlock trace) to understand the reasons why

the model contains an error. To find a counterexample for controllability is hard due to the criterion's nature. Controllability is "proven" by constructing a witness: N is k-controllable iff there *exists* some open net M such that the composition $N \oplus M$ is k-message bounded and weakly terminates. In other words, M can be seen as a counterexample for N's *un*controllability. If N is not controllable, we can only conclude that *no* such open net exists, and hence cannot provide a counterexample which can be used to find out which of the various problems we described in the last section rendered the net uncontrollable.

The algorithm to decide controllability (see Def. 5–6) overapproximates a partner for N and then iteratively removes states of this overapproximation that will not be part of any partner of N. If N is uncontrollable, all states will be eventually deleted. In this section, we define an algorithm to use information why states are deleted from $\mathcal{TS}_0$ to give diagnosis information for an uncontrollable net N.

As a motivation for the desired style of diagnosis information, consider again the open net in Fig. 2(e). We already described the reason why this net is uncontrollable: after sending an m-message, a partner has to either send an n-message or an o-message to the net. Depending on the net's choice, a sent message might keep pending on its channel. This eventually yields a covered final marking. Let's analyze this informal description of why the net is uncontrollable: it contains of:

(I) an indisputable initial part ("after sending an m-message"),
(C) a description of continuations ("a partner has to either send an n-message or an o-message"), and
(P) the problem that hinders a partner to control the net to a final marking ("Depending on the net's choice, a sent message might keep pending on its channel. This eventually yields a covered final marking.").

The initial part (I) consists of communication steps that are necessary to resolve a resolvable trapped marking and that would also be taken by partner who *knows* the outcome of the net's decision in advance. Sending m is not source of the problem, because m *will* be received by the net. In contrast, after sending m, any continuation (C) *can* lead to a situation where reaching a final marking is not any more guaranteed. Finally, the possible problem that can occur if after sending either message is described (P).

In the following, we generalize this approach and define an algorithm that automatically derives such diagnosis results for an uncontrollable net N consisting of these three parts:

(I) From $\mathcal{TS}_0$, we find a maximal subgraph $\mathcal{TS}_0^*$ such that the composition $N \oplus \mathcal{TS}_0^*$ is free of markings from which markings with exceeded message bound (EMB), deadlocks/livelocks (DLL), or covered final markings (CFM) cannot be avoided any more. We use $\mathcal{TS}_0$ as a starting point in order to consider all possible markings of N that can be reached with communication.
(C) The subgraph $\mathcal{TS}_0^*$ is not a partner of N, because its nodes contain resolvable trapped marking that are not resolved in $\mathcal{TS}_0^*$ (because the respective edge to

a successor is missing). When these resolvable trapped markings are resolved by sending messages to N, the composition might reach a state from which EMB, DLL, or CFM cannot be avoided any more. Therefore, in the second part of the diagnosis result, each unresolved resolvable trapped marking is described including a communication trace from q_0 to the state containing this resolvable trapped marking.

(P) Finally, we give detailed information how the resolution of the resolvable trapped marking can reach EMB, DLL, or CFM in the composition. For each problem, witness paths to the problematic situation and/or pointers to the structure of N are given to locate the problem.

To derive these diagnosis information — our counterexample for controllability —, we first need a criterion to decide for each state of $\mathcal{TS}_0$ whether it is a state of the subgraph $\mathcal{TS}_0^*$, too. We already motivated that $\mathcal{TS}_0^*$ should only contain those states which only contain markings from which it is still possible to avoid EMB, DLL, and CFM problems. For each problem, it is possible to characterize situations in which the problem either already occurred (e. g., an unresolvable trapped marking is reached) or cannot be avoided any more (e. g., only one transition is activated whose firing results in an an unresolvable trapped marking). If such a situation is found in a state of $\mathcal{TS}_0$, this state is obviously problematic. Thus, we define a blacklist for each problem that contains such problematic states. With these blacklists, we then can define the subgraph $\mathcal{TS}_0^*$.

In addition, for each problem, we define a witness. A witness is an artifact that can help to visualize the problem (e. g., a trace to an internal deadlock that can be simulated) or to locate the parts of the uncontrollable net that cause the problem (e. g., the transitions modeling a hidden choice). In the next three subsections, we define a blacklist and a witness for each problem (EMB, DLL, or CFM). Finally, we sketch an algorithm that applies the blacklists to $\mathcal{TS}_0$, analyzes the resulting subgraph $\mathcal{TS}_0^*$, and provides diagnosis information.

4.2 Blacklist for Deadlocking and Livelocking Control Flow

Unresolvable trapped markings and covered final markings have the same property: once the composition $N \oplus \mathcal{TS}_0$ reaches such a situation, a final marking becomes unreachable. Therefore, Def. 6 removes such states, because they jeopardize weak termination. However, the nature of these problems differ.

Unresolvable trapped markings model problems related to the control flow of N (modeled by $\mathit{Inner}(N)$). Therefore, detection and diagnosis of such control flow-related problems should differ from problems where the control flow of N already reached a final state, but an unreceived message is still pending on a channel. Hence, we differentiate unresolvable trapped markings from covered final markings.

We use the inner structure of N to detect markings from which a final marking is already unreachable. Any state of $\mathcal{TS}_0$ is blacklisted if it contains a marking from which, when restricted to the inner of N, a final marking is unreachable:

Definition 7 (DLL-blacklist, DLL-witness). *Define the* blacklist for a control flow deadlocks and livelocks *as* $blacklist_{DLL} = \{q \in Q_0 \mid \exists m \in q : m \text{ is an unresolvable trapped marking}\}$.

For each state $q^* \in blacklist_{DLL}$, *define* $\sigma_{DLL}(q^*)$ *to be a witness path* $N \oplus \mathcal{TS}_0$ *with* $[m_0, q_0] \xrightarrow{\sigma_{DLL}(q^*)} [m^*, q^*]$ *where* $m^* \in q^*$ *is an unresolvable trapped marking.*

In case m^* is a deadlock, the path $\sigma_{DLL}(q^*)$ is a witness path from the initial state of the composition to this deadlock. In case m^* is part of a livelock, any marking reachable from m^* is part of the same TSCC on which a final marking is unreachable and $\sigma_{DLL}(q^*)$ is a witness path for this TSCC.

4.3 Blacklist for Exceeded Message Bound

Markings that exceed the message bound k can be easily detected by analyzing the markings occurring in states of $\mathcal{TS}_0$. The blacklist as well as the witness can be defined straightforwardly:

Definition 8 (EMB-blacklist, EMB-witness). *Define the* blacklist for exceeded message bound *as* $blacklist_{EMB} = \{q \in Q_0 \mid \exists m \in q : \exists p \in I : m(p) > k\}$. *For each state* $q^* \in blacklist_{EMB}$, *define* $\sigma_{EMB}(q^*)$ *to be a witness path in* $N \oplus \mathcal{TS}_0$ *such that* $[m_0, q_0] \xrightarrow{\sigma_{EMB}(q^*)} [m^*, q^*]$ *with* $m^* \in q^*$ *with* $m^*(p) > k$.

The witness consists of a path in the composition that shows how the message bound of a place can be exceeded.

4.4 Blacklist for Covered Final Markings

In a covered final marking m_c reachable in $N \oplus \mathcal{TS}_0$, the control flow (the inner structure of N) has reached a final marking, but a message is pending on an input channel. Due to the construction of $\mathcal{TS}_0$ (cf. Def. 5(b)), this message was originally sent to N in order to resolve an resolvable trapped marking.

The following observation is needed to justify the later definition of a blacklist for covered final markings.

Lemma 1 (For each covered final marking there exists a final marking). *Let* N *be an open net and* $\mathcal{TS}_0$ *as defined in Def. 5. Let* q_1 *be a state of* $\mathcal{TS}_0$ *with a covered final marking* $m_1 \in q_1$ *with* $m_1 = m_f + m_i + [x]$ *such that* $m_f \in \Omega$ *is a final marking,* m_i *is a marking such that* $m_i(p) > 0$ *implies* $p \in P_{in}$, *and* $[x]$ *is a marking that marks an input place* $x \in P_{in}$. *Then exists a state* q_2 *of* $\mathcal{TS}_0$ *containing a marking* $m_2 \in q_2$ *with* $m_2 = m_f + m_i$.

Proof. Let q_e and q_x be states of $\mathcal{TS}_0$, let $[q_e, x, q_x]$ be an edge of $\mathcal{TS}_0$, and let σ be a path from q_x to q_1 that does not contain an x-labeled edge. Let $m_1 = m_f + m_i + [x]$ be as above. The x-message is only sent to N in order to resolve an resolvable trapped marking (cf. Def. 5). Let $m_e \in q_e$ be such a resolvable trapped marking.

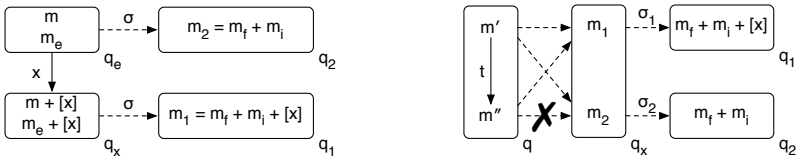


Fig. 3. Illustration of Lemma 1 (left) and a hidden choice transition of Def. 9 (right).

Let $m \in q_e$. Then $(m + [x]) \in q_x$. Let σ^* be an extension of σ to the composition $N \oplus \mathcal{TS}_0$ such that $[(m + [x]), q_x] \xrightarrow{\sigma^*} [(m_f + m_i + [x]), q_1]$. This transition sequence σ^* does not contain a transition producing a token on x, because $\sigma^*_{|\mathcal{TS}_0} = \sigma$ does not contain an x-labeled edge. Therefore, σ^* is realizable independently from the submarking $[x]$ of N; that is, σ^* is realizable *without* the submarking $[x]$. In particular, there exists a state q_2 of $\mathcal{TS}_0$ such that $[m, q_e] \xrightarrow{\sigma^*} [(m_f + m_i), q_2]$. □

Lemma 1 states that, for each covered final marking with a pending x-message occurring in a state of $\mathcal{TS}_0$, there exists a state that contains a covered final marking (or a final marking if $m_i = [\,]$) *without* that pending message. After iteratively applying Lemma 1, we can conclude that with each covered final marking occurring in $\mathcal{TS}_0$, also a respective "uncovered" final marking is present in a state of $\mathcal{TS}_0$.

Each application of Lemma 1 identifies an x-labeled edge from q_e to state q_x from which a state q_1 is reached that contains a covered final marking with a pending x-message. For state q_e, an alternative continuation to q_2 without an x-edge is possible. Figure 3(a) illustrates this.

Hence, such a state q_x should be considered critical which yields the following definition of a blacklist for covered final markings:

Definition 9 (CFM-blacklist, CFM-witness). *Define the* blacklist for covered final markings*, $blacklist_{CFM}$, to contain all states $q_x \in Q_0$ such that:*

- *there exists a state q_1 and a path σ in $\mathcal{TS}_0$ with $q_x \xrightarrow{\sigma} q_1$,*
- *$m_c \in q_1$ is a covered final marking with $m_c(x) > 0$ for an input place $x \in P_{in}$*
- *σ does not contain an x-labeled edge*
- *q_e is a predecessor of q_x with an edge $[q_e, x, q_x]$*

For each state $q_x \in blacklist_{CFM}$, define $\sigma_{CFM}(q_x) = [\sigma_1, \sigma_2, T^*]$ to be a witness where:

- *σ_1 is a path in $N \oplus \mathcal{TS}_0$ with $[m_1, q_x] \xrightarrow{\sigma_1} [(m_f + m_i + [x]), q_1]$ where $m_1 \in q_x$,*
- *σ_2 is a path in $N \oplus \mathcal{TS}_0$ with $[m_2, q_x] \xrightarrow{\sigma_2} [(m_f + m_i), q_2]$ where $m_2 \in q_x$ such that $\sigma_{1|N} = \sigma_{2|N}$, and*
- *the set T^* containing all transitions $t \in T$ of N such that:*
  - *there exists a state q in $\mathcal{TS}_0$ with $m', m'' \in q$,*
  - *$[m', q] \xrightarrow{*} [m_1, q_x]$, $[m', q] \xrightarrow{*} [m_2, q_x]$, $[m', q] \xrightarrow{t} [m'', q]$,*
  - *$[m'', q] \xrightarrow{*} [m_1, q_x]$, and $[m'', q] \not\xrightarrow{*} [m_2, q_x]$.*

Algorithm 1 Blacklist-based diagnosis for uncontrollable open nets

Input: uncontrollable, normal, and bounded open net N; message bound $k \in \mathbb{N}^+$
Output: diagnosis information why N is uncontrollable; subgraph $\mathcal{TS}_0^*$ of $\mathcal{TS}_0$

calculate $\mathcal{TS}_0$ from N
calculate $blacklist_{DLL}$ from $Inner(N)$
calculate $blacklist_{EMB}$ from $\mathcal{TS}_0$
calculate $blacklist_{CFM}$ from $\mathcal{TS}_0$

if q_0 is blacklisted **then**

  if $q_0 \in blacklist_{DL}$ **then**
    print "control flow deadlock or livelock reachable without interaction"
    print DL-witness $\sigma_{DL}(q_0)$

  if $q_0 \in blacklist_{EMB}$ **then**
    print "message bound of communication place p exceeded without interaction"
    print EMB-witness $\sigma_{EMB}(q_0)$

else

  for all states q reachable from q_0 by a sequence σ without visiting blacklisted states **do**
    if (state q is not blacklisted **and** no resolvable trapped marking in q is resolved) **then**
      for all resolvable trapped markings $m_e \in q$ where
        ($m_{e|Inner(N)}$ activates a transition t with $\ell(t) = x$ **and**
        state q_x reachable by edge $[q, x, q_x]$ is blacklisted) **do**
        print path σ from q_0 to q
        if $q_x \in blacklist_{DLL}$ **then**
          print "sending message x to N to resolve an resolvable trapped marking in q_e reachable by σ from q_0 can reach a control flow deadlock or livelock"
          print DLL-witness $\sigma_{DLL}(q_x)$
        if $q_x \in blacklist_{EMB}$ **then**
          print "sending message x to N to resolve an resolvable trapped marking in q_e reachable by σ from q_0 can exceed the message bound"
          print EMB-witness $\sigma_{EMB}(q_x)$
        if $q_x \in blacklist_{CFM}$ **then**
          print "message x necessary to resolve an resolvable trapped marking in q_e reachable by σ from q_0 might be left unreceived due to a hidden choice"
          print CFM-witness $\sigma_{EMB}(q_x)$

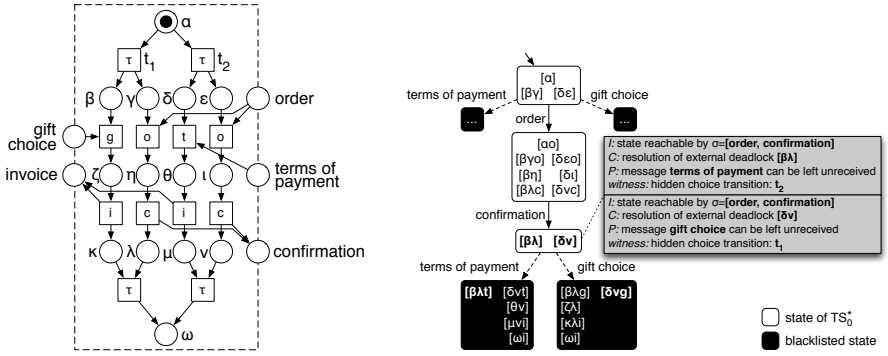
  print subgraph $\mathcal{TS}_0^*$ of $\mathcal{TS}_0$ where all blacklisted states are removed

A covered final marking is a situation that occurs when N is composed to a partner. With the help of Lemma 1, the blacklist for covered final markings can be defined only by checking markings of nodes of $\mathcal{TS}_0$ and paths in $\mathcal{TS}_0$. This can be easily done while building $\mathcal{TS}_0$ instead of analyzing paths in $N \oplus \mathcal{TS}_0$. Lemma 1 also allows for finding a set T^* of *hidden choice transitions* (see Fig. 3(b)) which model a hidden decision as described in Sect. 3.2. These transitions can be the starting point to repair the net to avoid the covered final marking.

4.5 Blacklist-based Diagnosis

With the definitions of the blacklists for deadlocks/livelocks, exceeded message bound, and covered final markings, we are able to define the subgraph (i. e., the counterexample for controllability of N) $\mathcal{TS}_0^*$ of $\mathcal{TS}_0$ which only contains states that are not contained in any of the three blacklists.

Algorithm 1 combines the defined blacklists together with their witnesses, and gives information for each detected problem. After a preprocessing phase



(a) online shop model (b) counterexample for controllability

Fig. 4. Example for the application of Algorithm 1.

(line 1–4) in which $\mathcal{TS}_0$ as well as the blacklists are calculated, the states of $\mathcal{TS}_0$ are analyzed. Thereby, two cases are differentiated: if already the initial state of q_0 is blacklisted, then the open net reaches a problem independently of a partner. Covered final markings cannot occur in this setting. As a diagnosis information, the initial state q_0 and the respective problem(s) are printed (line 5–11). The rest of the algorithm (line 12–26) treats situations in which $\mathcal{TS}_0^*$ is nonempty.

The diagnosis messages can be classified into the three categories (initial part `I`, possible continuation `C`, and occurring problem `P`) as follows:

(`I`) line 26 prints the non-blacklisted subgraph $\mathcal{TS}_0^*$,
(`C`) line 16 prints, for each resolvable trapped marking that is not resolved in $\mathcal{TS}_0^*$, a communication trace from q_0 to the state containing the resolvable trapped marking,
(`P`) line 7–8, 10–11, 18–19, 21–22, and 24–25 print information about the problem that might be unavoidable after resolving the respective deadlock, including witnesses.

It is worth mentioning that the algorithm lists all problems that can occur if $\mathcal{TS}_0^*$ is "left" by resolving a deadlock. If, for example, sending an x-message in some state can result in a message bound violation *and* yield an internal deadlock, then both problems are reported.

4.6 Diagnosis Example

To demonstrate the proposed diagnosis framework, we applied Algorithm 1 to an uncontrollable net we presented and discussed in [20]. This net which is depicted in Fig. 4(a) is a model of a WS-BPEL online shop which after receiving a login message from a customer (not depicted in Fig. 4), internally decides whether to treat the customer as premium customer (t_1) or standard customer (t_2). As the customer is not informed about the outcome of the decision, the shop is uncontrollable, because either a message gift choice or terms of payment might be

left unreceived. Experiments showed that even experienced Web service designers would consider the online shop as correct even when it is presented in other formalisms such as WS-BPEL or BPMN. This underlines the need for a diagnosis framework for such incorrect models, because concurrent control flow combined with asynchronous message flow is hard to oversee.

Figure 4(b) shows the result of Algorithm 1 when applied to the online shop model. It discovers the hidden choice modeled by transitions t_1 and t_2. These transitions can be used as a starting point to fix the online shop. If, for example, the outcome of the internal decision would be communicated to the customer (e. g., by adding two output places premium and standard which are connected to t_1 and t_2, resp.), the shop would be controllable.

4.7 Implementation of the Diagnosis Algorithm

The diagnosis algorithm is based on the partner overapproximation $\mathcal{TS}_0$ which can be infinite if the net under consideration has no message bound (e. g., Fig. 2(a)). We already required that a message bound k is given for the analysis, so instead of first calculating $\mathcal{TS}_0$ and then removing states with exceeded message bound, $\mathcal{TS}_1$ can be calculated in the first place. In fact, this is exactly the way the controllability decision algorithm is implemented in the tool Fiona [20].

The diagnosis algorithm can be implemented straightforwardly. The set $\mathit{blacklist}_{DLL}$ can be calculated in a preprocessing stage. During the calculation of $\mathcal{TS}_1$, $\mathit{blacklist}_{EMB}$ can be built whenever the currently calculated state contains a marking with exceeded message bound. Whenever a state with a covered final marking is detected, $\mathit{blacklist}_{CFM}$ can be filled according to the criteria of Lemma 1.

5 Conclusion

The concept of counterexamples greatly boosted the acceptance of model checking [14] in the field of computer-aided verification. However, the decision algorithm for controllability [3, 15] does not give such counterexamples in case of uncontrollability. In this paper, we investigated uncontrollable service models and presented a variety of reasons why a service does not have any partners that interact deadlock and livelock freely. We presented an algorithm to analyze an uncontrollable service model to give diagnosis information *why* the model is uncontrollable. This diagnosis information can be a starting point for corrections [4] of the model towards controllability to meet this fundamental correctness criterion for service-oriented computing.

In future work, we plan to implement the diagnosis algorithm in the tool Fiona [20]. Fiona already implements some reduction techniques [21] to efficiently decide controllability. We will investigate whether these techniques can be combined with the diagnosis algorithm. As service models usually stem from industrial specification languages such as WS-BPEL, the retranslation of (Petri net-related) diagnosis information back into WS-BPEL is subject of future work. Together with the translations [9, 10] from and to WS-BPEL, the diagnosis algorithm can be directly applied to industrial service specification languages.

Acknowledgements The author wishes to thank Karsten Wolf for advice on the relationship to classical model checking and the anonymous referees for their valuable comments. Niels Lohmann is funded by the DFG project "Operating Guidelines for Services" (WO 1466/8-1).

References

1. Papazoglou, M.P.: Agent-oriented technology in support of e-business. Commun. ACM **44**(4) (2001) 71–77
2. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures and Applications. Springer (2003)
3. Schmidt, K.: Controllability of open workflow nets. In: EMISA 2005. Volume P-75 of LNI., GI (2005) 236–249
4. Lohmann, N.: Correcting deadlocking service choreographies using a simulation-based graph edit distance. In: BPM 2008. Volume 5240 of LNCS., Springer (2008) 132–147
5. Aalst, W.M.P.v.d.: The application of Petri nets to workflow management. Journal of Circuits, Systems and Computers **8**(1) (1998) 21–66
6. Verbeek, H.M.W., Basten, T., Aalst, W.M.P.v.d.: Diagnosing workflow processes using Woflan. Comput. J. **44**(4) (2001) 246–279
7. Massuthe, P., Reisig, W., Schmidt, K.: An operating guideline approach to the SOA. Annals of Mathematics, Computing & Teleinformatics **1**(3) (2005) 35–43
8. Reisig, W.: Petri Nets. EATCS Monographs on Theoretical Computer Science edn. Springer (1985)
9. Lohmann, N.: A feature-complete Petri net semantics for WS-BPEL 2.0. In: WS-FM 2007. Volume 4937 of LNCS., Springer (2008) 77–91
10. Lohmann, N., Kleine, J.: Fully-automatic translation of open workflow net models into simple abstract BPEL processes. In: Modellierung 2008. Volume 127 of LNI., GI (2008) 57–72
11. Alves, A., et al.: Web Services Business Process Execution Language Version 2.0. OASIS Standard, 11 April 2007, OASIS (2007)
12. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: ICATPN 2007. Volume 4546 of LNCS., Springer (2007) 321–341
13. Massuthe, P., Serebrenik, A., Sidorova, N., Wolf, K.: Can I find a partner? Undecidablity of partner existence for open nets. Inf. Process. Lett (2008) (Accepted for publication).
14. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press (2000)
15. Wolf, K.: Does my service have partners? LNCS ToPNoC (2008) (Accepted for publication).
16. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge University Press (1995)
17. Dehnert, J., Aalst, W.M.P.v.d.: Bridging the gap between business models and workflow specifications. Int. J. Cooperative Inf. Syst. **13**(3) (2004) 289–332
18. Lautenbach, K., Ridder, H.: Liveness in bounded Petri nets which are covered by T-invariants. In: ATPN 1994. Volume 815 of LNCS., Springer (1994) 358–375
19. Lohmann, N., Massuthe, P., Wolf, K.: Behavioral constraints for services. In: BPM 2007. Volume 4714 of LNCS., Springer (2007) 271–287
20. Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing interacting WS-BPEL processes using flexible model generation. Data Knowl. Eng. **64**(1) (2008) 38–54
21. Weinberg, D.: Efficient controllability analysis of open nets. In: WS-FM 2008. LNCS, Springer (2008)