Soundness-preserving refinements of service compositions

Soundness-Preserving Refinements of Service Compositions

Kees M. van Hee, Arjan J. Mooij, Natalia Sidorova, and Jan Martijn van der Werf

Department of Mathematics and Computer Science*,
Technische Universiteit Eindhoven, The Netherlands
{K.M.v.Hee,A.J.Mooij,N.Sidorova,J.M.E.M.v.d.Werf}@tue.nl

Abstract. Soundness is one of the well-studied properties of processes; it denotes that a final state can be reached from every state that is reachable from the initial state. Soundness-preserving refinements are important for enabling the compositional design of systems.

In this paper we concentrate on refinements of service compositions. We model service compositions using Petri nets, and consider specific pairs of places that belong to different services. Starting from a sound service composition, we show how to check whether such a pair of places can be refined by another sound service composition, so that soundness is preserved through the refinement.

Keywords: Service composition, refinement, Petri net, soundness, verification.

1 Introduction

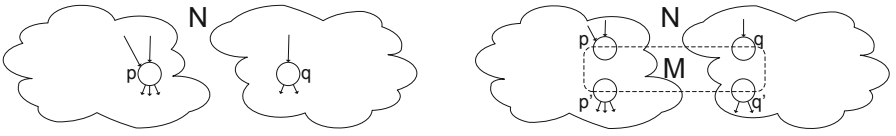
Recent developments such as component-based software engineering (CBSE) and service-oriented architectures (SOA) have led to systems that are composed from many services. Each service delivers a specific functionality, and communicates asynchronously with some other services using messages. In turn, a service itself may be composed out of several other (communicating) services, resulting in an intricate network of services.

This trend became only more visible with the adoption of the Software as a Service (SaaS) paradigm that facilitates the communication across boundaries of organizations. As a consequence, it became virtually impossible for a single organization to obtain a full model of the system, and hence it became even more challenging to ensure its (behavioral) correctness.

In this paper we study compositional design methods that ensure correctness of service compositions based on properties of communicating pairs of services. One of the main formalisms for modeling and analyzing systems that communicate asynchronously are Petri nets, which we use in this paper. The behavioral

* Author Mooij participates in the Poseidon project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

M. Bravetti and T. Bultan (Eds.): WS-FM 2010, LNCS 6551, pp. 131–145, 2011.



(a) Net N (b) Net $N \odot M$

Fig. 1. Refinement of (synchronizable) places

correctness property that we consider is *soundness* [1], or *weak termination*, which requires that a final state can be reached from every state that is reachable from the initial state. Soundness has been studied extensively, and has proved to be practically relevant.

Compositional techniques for design and analysis have a long tradition. A nice overview of fundamental refinement techniques for Petri nets is given in [4,29]. In the context of component-based systems and service-oriented technologies, a lot of research considers communicating systems, see e.g., [14,26,27], in combination with some variants of soundness. However, these works focus on "horizontal" modularization (i.e., composition) of communicating systems, while we are interested here in "vertical" modularization (i.e., refinement). Conditions for vertical modularization were given by [28,12], regarding soundness-preserving refinements of a single place (in a Petri net).

In this paper we consider the refinement of a pair of places p and q in a Petri net N by a sound workflow net M with two designated initial (source) places p and q and two final (sink) places p' and q'; see Fig. 1. Both net N and net M model service compositions that may involve multiple communicating services. We define conditions for refined net N and refining net M in isolation such that the refinement is sound.

Overview. In Section 2, we summarize the basic definitions related to Petri nets and the accordance relation. In Section 3, we introduce the refinement of synchronizable places and give the intuition behind this concept. In Section 4, we present a homogeneous criterion for refinement based on two soundness checks. In Section 5 we formally prove its correctness. We conclude in Section 6 with some conclusions and further work.

2 Preliminaries

Let S be a set. A *sequence* σ of length $n \in \mathbb{N}$ over S is a function $\sigma : \{1, \ldots, n\} \to S$; we denote its length by $|\sigma| = n$. If $|\sigma| = 0$, then σ is the *empty sequence* ϵ. The set of all finite sequences over S is denoted by S^*.

A *bag* (or *multiset*) m over S is a function $m : S \to \mathbb{N}$. We use '.' to denote function application; so, for $s \in S$, $m.s$ denotes the number of occurrences of s in m. We write $\mathbb{N}^S$ for the set of all bags over S, and $[s]$ for the bag containing one occurrence of $s \in S$. We use $+$ for the sum of two bags, and $\leq$ for the comparison of two bags. Sets can be seen as bags in which all elements have multiplicity one.

Let $\mathcal{A}$ be the universe of actions, not including silent (or internal) action τ.

Definition 1 (Labeled transition system). *A* labeled transition system*, LTS for short, L is a 4-tuple $(S, \longrightarrow, s_i, \Omega)$, where S is a set of* states*; $\longrightarrow \subseteq S \times (\mathcal{A} \cup \{\tau\}) \times S$ is a* transition relation*; $s_i \in S$ is the* initial state*; and $\Omega \subseteq S$ is the set of* final states*.*

For $s, s' \in S$ and $a \in \mathcal{A}$, we write $s \xrightarrow{a} s'$ if and only if $(s, a, s') \in \longrightarrow$. A state $s \in S$ is called a *deadlock* if no action $a \in \mathcal{A} \cup \{\tau\}$ and state $s' \in S$ exist such that $s \xrightarrow{a} s'$. If for some $\sigma \in (\mathcal{A} \cup \{\tau\})^*$ of length n, and states $s_i \in S$ for $0 \leq i \leq n$ such that $s_{i-1} \xrightarrow{\sigma.i} s_i$ for $0 < i \leq n$, we write $s_0 \xrightarrow{\sigma} s_n$. The set of reachable states from a given state $s \in S$ is defined as $\mathcal{R}(L, s) = \{s' \in S \mid \exists \sigma \in (\mathcal{A} \cup \{\tau\})^* : s \xrightarrow{\sigma} s'\}$.

Definition 2 (Weak termination). *An LTS L is* weakly terminating *if for every state $s \in \mathcal{R}(L, s_i)$, $\Omega \cap \mathcal{R}(L, s) \neq \emptyset$ holds.*

2.1 Petri Nets, Workflows and Soundness

A *Petri net* is a 3-tuple $N = (P, T, F)$, where P and T are two disjoint sets of *places* and *transitions* respectively; and $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation*. The elements from the set $P \cup T$ are called the *nodes* of N. Elements of F are called *arcs*. Given a node $n \in (P \cup T)$, we define its *preset* ${}^\bullet n = \{n' \mid (n', n) \in F\}$, and its *postset* $n^\bullet = \{n' \mid (n, n') \in F\}$. Graphically, places are depicted as circles, transitions as squares, and arcs as arrows.

Markings are the states of a net; each *marking* m of a Petri net $N = (P, T, F)$ is a bag over P. A transition t is *enabled* in a marking m if ${}^\bullet t \leq m$; *firing* an enabled transition t in marking m yields a marking m' such that $m' + {}^\bullet t = m + t^\bullet$.

A *system* is a triple (N, m, Ω), where N is a Petri net, $m \in \mathbb{N}^P$ is the initial marking and $\Omega \subseteq \mathbb{N}^P$ is a set of final markings. The semantics of a system (N, m, Ω) is defined as an LTS $(\mathbb{N}^P, \longrightarrow, m, \Omega)$, where $(m, t, m') \in \longrightarrow$ if and only if $m' + {}^\bullet t = m + t^\bullet$ and ${}^\bullet t \leq m$.

Workflow nets are special Petri nets with one initial place and one final place.

Definition 3 (Workflow net, soundness). *A Petri net (P, T, F) is called a* workflow net *if (1) there exists exactly one place $i \in P$, called the* initial place*, such that ${}^\bullet i = \emptyset$, (2) there exists exactly one place $f \in P$, called the* final place*, such that $f^\bullet = \emptyset$, and (3) all nodes are on a path from i to f.*

A workflow net N is sound *if the LTS semantics of system $(N, [i], \{[f]\})$ is weakly terminating.*

In the temporal logic CTL (Computation Tree Logic, [9]), weak termination (and hence soundness) can be expressed using the "AG EF" pattern, where AG refers to every reachable state, and EF refers to the existence of a (terminating) path. Such properties can be checked for Petri nets using tools like LoLA [25].

2.2 Open Nets and Composition

We use an extension of Petri nets that is called open nets [13,18]. To model external asynchronous communication, open nets have an interface that consists of input places and output places.

Definition 4 (Open (Petri) net, soundness). *An* open (Petri) net *is a* 7*-tuple* $(P, T, F, P_i, P_o, m_0, \Omega)$*, where* $((P, T, F), m_0, \Omega)$ *is a system;* $P_i \subseteq P$ *is a set of input places such that* ${}^\bullet p = \emptyset$ *for all* $p \in P_i$*;* $P_o \subseteq P$ *is a set of output places such that* $p^\bullet = \emptyset$ *for all* $p \in P_o$*; and* $m.p = 0$ *for all markings* $m \in \Omega \cup \{m_0\}$ *and places* $p \in P_i \cup P_o$*.*

A closed net *is an open net without asynchronous interface places, i.e.,* $P_i = P_o = \emptyset$*. A closed net* N *is called* sound*, denoted by SD.*N*, if the LTS semantics of its system is weakly terminating.*

To model synchronous communication, we extend open nets as in [22] with a total labeling function L, which assigns to every transition a label that denotes the synchronous port (if any) it is connected to. If a transition is not connected to any synchronous port, it is assigned the auxiliary label τ. A closed net also has no synchronous interface ports, i.e., $(\forall t : t \in T : \ L.t = \tau)$.

Traditional open nets (without synchronous ports) can be composed by fusing interface places that are an input place of one net and an output place of the other, resulting in internal places. Two nets are composable if and only if their shared interface places are of this kind. Open nets with synchronous ports can be composed by also fusing each pair of transitions from the two nets with identical non-τ labels, resulting in τ-labeled transitions.

Without loss of generality, we assume that all nodes except the interfaces of the involved nets are disjoint, which can be achieved by renaming the internal (non-interface) places and transitions.

Definition 5 (Composition, [22]). *Let* N_1 *and* N_2 *be open nets.* N_1 *and* N_2 *are* composable *iff* $P_{i1} \cap P_{i2} = \emptyset$ *and* $P_{o1} \cap P_{o2} = \emptyset$*. If* N_1 *and* N_2 *are composable, their* composition $N = N_1 \oplus N_2$ *is defined by*

$$P = P_1 \cup P_2; \ \ P_i = (P_{i1} \cup P_{i2}) \setminus (P_{o1} \cup P_{o2}); \ \ P_o = (P_{o1} \cup P_{o2}) \setminus (P_{i1} \cup P_{i2});$$
$$\Omega = \{m_1 + m_2 \mid m_1, m_2 : \ m_1 \in \Omega_1 \ \wedge \ m_2 \in \Omega_2\}; \quad m_0 = m_{01} + m_{02};$$

$$T = T_f \cup T_s;$$
$$T_f = \{t \mid t : \ t \in T_1 \ \wedge \ (L_1.t = \tau \ \vee \ (\forall t' : t' \in T_2 : L_1.t \neq L_2.t'))\}$$
$$\cup \ \{t \mid t : \ t \in T_2 \ \wedge \ (L_2.t = \tau \ \vee \ (\forall t' : t' \in T_1 : L_2.t \neq L_1.t'))\};$$
$$T_s = \{\{t_1, t_2\} \mid t_1, t_2 : \ t_1 \in T_1 \ \wedge \ t_2 \in T_2 \ \wedge \ L_1.t_1 = L_2.t_2 \ \wedge \ L_1.t_1 \neq \tau\};$$

$$L = \{[t, (L_1 \cup L_2).t] \mid t : \ t \in T_f\} \ \cup \ \{[\{t_1, t_2\}, \tau] \mid t_1, t_2 : \ \{t_1, t_2\} \in T_s\};$$

$$F = ((F_1 \cup F_2) \ \cap \ ((P \cup T_f) \times (P \cup T_f)))$$
$$\cup \ \{[p, \{t_1, t_2\}] \mid p, t_1, t_2 : \ \{t_1, t_2\} \in T_s \ \wedge \ ([p, t_1] \in F_1 \ \vee \ [p, t_2] \in F_2)\}$$
$$\cup \ \{[\{t_1, t_2\}, p] \mid p, t_1, t_2 : \ \{t_1, t_2\} \in T_s \ \wedge \ ([t_1, p] \in F_1 \ \vee \ [t_2, p] \in F_2)\}.$$

2.3 Accordance

A *controller* of an open net S is an open net R such that $SD.(R \oplus S)$ holds. A *controllable* open net is an open net that has at least one controller.

Definition 6 (Accordance pre-order). *The accordance pre-order $\leq$ on open nets S and T is defined as:* $S \leq T \;\equiv\; (\forall R ::\; SD.(R \oplus T) \;\Rightarrow\; SD.(R \oplus S))$.

The accordance pre-order [3] is equivalent to the conflict pre-order from [17], and the sub-contract pre-order from [5]. In [20] the relation between this pre-order and other pre-orders (in particular, fair testing [24]) has been studied.

Definition 7 (Maximal controller, [21]). *A maximal controller of a controllable open net S is an open net $mc.S$ such that:*

$$(\forall R ::\; SD.(R \oplus S) \;\equiv\; R \leq mc.S)$$

We can simplify the two implications inside the equivalence $\equiv$ as follows:

$\Leftarrow$: $mc.S$ is a controller: $SD.(mc.S \oplus S)$, and
$\Rightarrow$: $mc.S$ is larger than all controllers: $(\forall R ::\; SD.(R \oplus S) \;\Rightarrow\; R \leq mc.S)$.

Using the maximal controller, we can decide accordance by checking deadlock-freedom in the composition of T and a maximal controller of S. Similar decision procedures for different pre-orders have been studied in [6,10].

Proposition 1 (Deciding accordance using maximal controller, [19]). *For each open net S and controllable open net T:* $S \leq T \;\equiv\; SD.(S \oplus mc.T)$.

3 Synchronizable Places

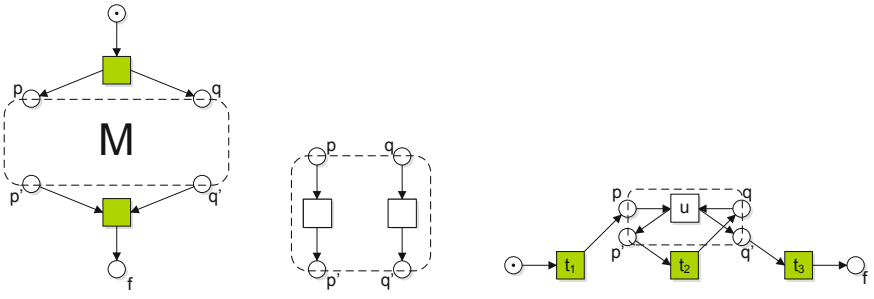
In this section we explore the problem of preserving soundness while refining pairs of (synchronizable) places. We apply a semi-formal style.

3.1 Introduction

Service compositions consist of several independent services that communicate via interfaces. We use Petri nets to model the communication between such services. Our aim is to support the design of nets in a top-down manner, in particular by refining pairs of places.

Suppose a closed net N is given, which contains two internal places p and q (which do not occur in the initial marking nor in any final marking). These two places can be refined by an open net M with input places p and q, and output places p' and q'. This refinement, denoted by $N \odot M$, is sketched in Fig. 1.

If places p and q in N are related to two different services, we thus impose the additional communication protocol M on the two services to which places p and q in N belong. We assume that open net M also models several services, i.e., input place p models the initial state of one service, and output place p'



(a) Workflow (b) Forward (c) Sequence

Fig. 2. Exploration: some (counter) examples

models the end state of the service; similarly for q and q' in relation to another service. Moreover, net M consumes a token from p before it produces a token in p'; similarly for places q and q'. Finally, we assume that the net in Fig. 2(a), which contains M, is a sound workflow.

In what follows, we study under which conditions p and q in net N can be called *synchronizable*, i.e., under which conditions the refinement $N \odot M$ is sound. Such conditions must implicitly approximate M, but independently of any specific M.

3.2 Soundness

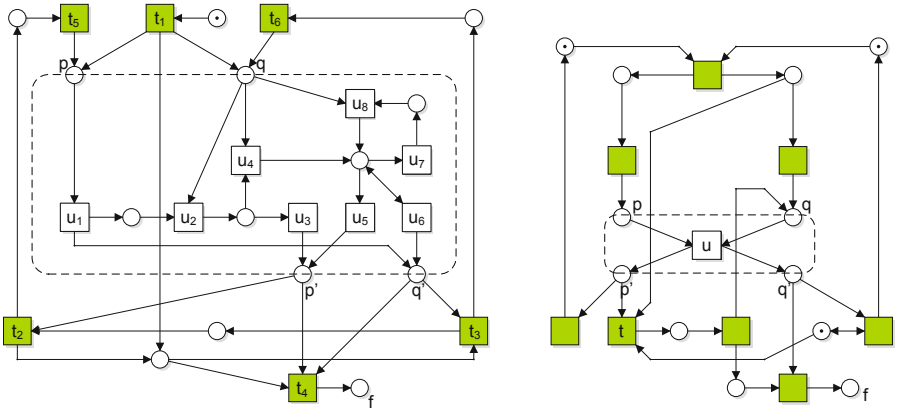
Soundness of N is a necessary condition for the refinement. Consider for example the net M in Fig. 2(b), for which $N \odot M$ is equivalent (fusion of series places [23]) to N, for every net N. For $N \odot M$ to be sound, net N must be sound.

However, this is not a sufficient condition. An example refinement $N \odot M$ is depicted in Fig. 2(c). Net N contains places p and q that are "sequentially connected" by transition t_2, and net M is a simple synchronizing net. Although N is sound, $N \odot M$ reaches a deadlock after firing transition t_1. From this example we conclude that it should be possible to mark places p and q in net N at the same time.

3.3 Refinement with the Simplest Synchronizing Net

The previous subsection suggests to check soundness of N refined with the simplest synchronizing net M, viz., a single transition, as depicted in Fig. 2(c).

However, this is not a sufficient condition. An example refinement $N \odot M$ is depicted in Fig. 3(a). Net N fires transition t_1, followed by an alternation between transitions t_3, t_6 and t_2, t_5, and finally transition t_4. Net N is sound, even after refinement with the simplest synchronizing net. However, in the refinement $N \odot M$, net M can consume a second token from q before producing a token in p' (or net N can produce a second token in q before consuming a token from p'). After firing transitions t_1, u_1 and u_2, transitions t_3 and t_6 can fire, and



(a) Deadlock after refinement (b) Livelock after refinement

Fig. 3. Exploration: more (counter) examples

hence transitions u_4 and u_7 can fire; thus resulting in a deadlock. The net is even unbounded, as after transitions t_1, u_1, u_2, t_3, t_6 and u_4, transition u_6 can fire unlimitedly. Net N enables behavior in net M that is not considered by the soundness check on M as the transitions u_4, u_5, u_6, u_7 and u_8 are dead.

Thus the simplest synchronizing net M as depicted in Fig. 2(c) is not a proper approximation of each sound workflow. For the AG-part of soundness, we conclude that net N should not contain transitions (like the ones we have just discovered) that may lead M into unexplored behavior.

3.4 Some Transitions Should Not Occur

The previous subsection suggests to check that N can only produce tokens on place p (using action p) and consume tokens from place p (using action p') in the orders described in Fig. 4. The initial state is s_0, and both the solid and the dashed transitions are permitted. For readability reasons, each state is annotated with the number of tokens in places p and q. Without loss of generality, we assume that each transition in N performs at most one action. In particular, in states s_4, s_5 and s_7, Fig. 4 excludes producing a token in q for the second time before any token has been consumed from p; thus excluding the example in Fig. 3(a).

However, this is not a sufficient condition. An example refinement $N \odot M$ is depicted in Fig. 3(b). In net N, transition t is crucial for termination; its firing implies that N has produced a token on p, but not yet on q. Net N only executes actions in the order specified in Fig. 4, but in $N \odot M$, net M eliminates the path to termination from net N. So M imposes additional synchronization on net N.

Thus the solid and dashed transitions in Fig. 4 are not a proper approximation of each sound workflow. For the EF-part of soundness, we conclude that net N should only use transitions that cannot be excluded by any M.

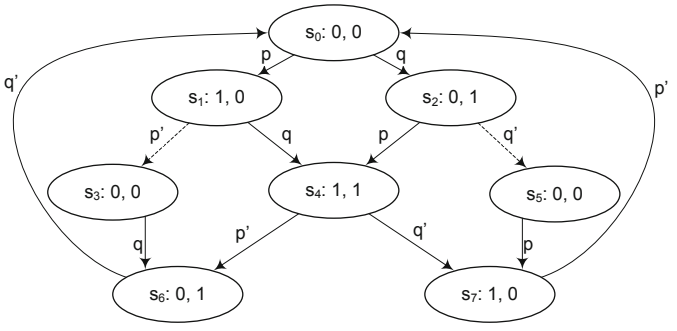


Fig. 4. May/exit transition system

3.5 Some Transitions Should Occur

The previous subsection suggests to check that also from every reachable non-final state of N, a final state can be reached using only the solid transitions in Fig. 4. In particular, in state s_1, this excludes that for termination of net N first a token from p must be consumed before any token on q has been produced.

Fig. 4 contains a may/exit transition system, where all transitions are may-transitions, and the solid transitions are also exit-transitions. Thus, we obtain a sufficient condition for synchronizable places p and q in N, viz., N should be sound under the restrictions from the may/exit transition system in Fig. 4. That is, for the AG-part of soundness both the exit- and the may-transitions can be used, whereas for the EF-part of soundness only the exit-transitions can be used.

In relation to modal (may/must) transition systems [15], the may-transitions correspond, whereas the exit- and must-transitions are unrelated. In relation to game theory [8], the may-transitions are "conserving", and the exit-transitions are "equalizing". A detailed study of may/exit transition systems is outside the scope of this paper.

4 Homogeneous Solution

The refinement described in Section 3 depends on a sound workflow for M and a special kind of soundness for N that takes into account the may/exit transition system from Fig. 4. In this section we show how these criteria can be formulated in a homogeneous way as two checks for (traditional) soundness. We start by defining place refinement in terms of composition of open nets.

4.1 Refinement in Terms of Composition

Suppose a net N is given with places p and q as in Fig. 5(a). To separate the incoming and outgoing arcs from these places we apply fusion of series places [23] and obtain Fig. 5(b). By definition of composition of asynchronous interfaces, this is equal to Fig. 5(c).

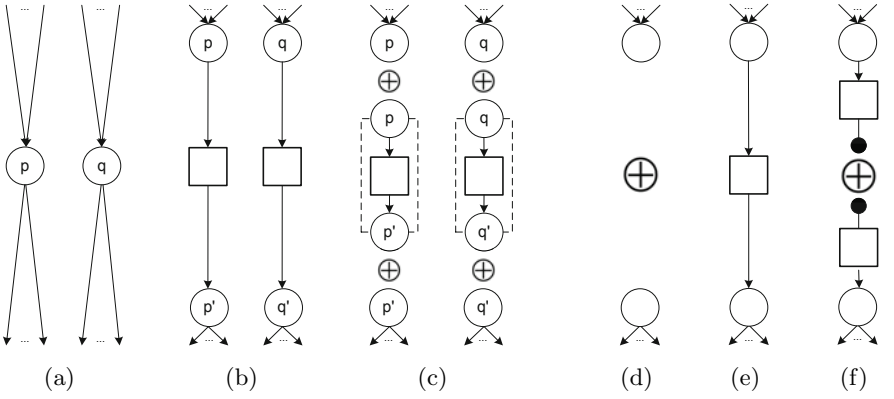


Fig. 5. Refinement in terms of composition

Thus each net N is equivalent to a similar open net N' composed with the open net from Fig. 2(b). The refinement of net N with an open net M (with an asynchronous interface), denoted by $N \odot M$, is defined as $N' \oplus M$.

Although asynchronous interfaces are most natural in Petri nets, our results are easier to explain in terms of synchronous interfaces. Therefore we show how two nets with an asynchronous interface can be translated into two nets with a synchronous interface, in such a way that the compositions are equivalent.

Consider any pair of corresponding interface places from the two nets as in Fig. 5(d). After composing them, we apply fusion of series places [23] and obtain Fig. 5(e). By definition of composition of synchronous interfaces, this is equal to Fig. 5(f). So every asynchronous interface place in the open nets N' and M becomes an internal place that is connected by a transition to a synchronous interface port (indicated by a black dot). Thus we transform open nets N' and M into open nets N'' and M'' such that $N \odot M$ is equivalent to $N'' \oplus M''$. In what follows, we use N and M to refer to N'' and M''.

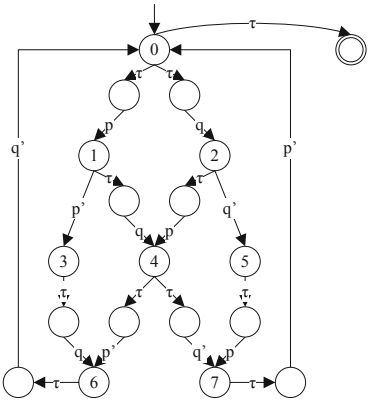
4.2 Two Checks for Soundness

In this section we propose a pair of open nets E (environment) and T (test) with the same synchronous interface as N and M. The idea is to conclude that $N \oplus M$ is sound if both $N \oplus T$ and $E \oplus M$ are sound:

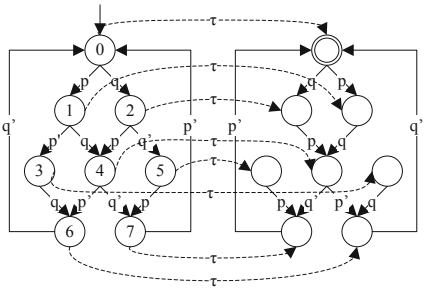
$$(\forall N, M :: \ SD.(N \oplus T) \ \wedge \ SD.(E \oplus M) \ \ \Rightarrow \ \ SD.(N \oplus M))$$

For brevity reasons, in Fig. 6 we describe open nets E and T using an LTS (where double circles denote final states) with some additional transitions; such an LTS can be translated to an open net with a synchronous interface (where every transition has one incoming and one outgoing arc). For every numbered state i and action a mentioned in Fig. 6(c), a transition $i \xrightarrow{a} b$ should be added (called a "fact"-transition in [11]), where state b is a deadlock state.

In the remainder of this section we motivate these nets E and T using the insights from Section 3. In Section 5 we formally show their correctness, and the way in which net T can be computed from net E.



(a) Environment E



(b) Test T

	0	1	2	3	4	5	6	7
E	p', q'	q'	p'	p', q'		p', q'	p'	q'
T		p	q	p	p, q	q	p, q	p, q

(c) Additional fact transitions

Fig. 6. State machines for the pair $\langle E, T \rangle$

Net E: The condition on M in Section 3 is that Fig. 2(a) yields a sound workflow. Definition 3 (Workflow) requires that all nodes are on a path from i to f, which, in combination with soundness, guarantees that after M has produced all output tokens, all non-output places of net M are empty. As the internal places of M cannot be accessed using composition, the overall structure of net E in Fig. 6(a) checks that M is sound even when executed multiple times. Note that this condition is slightly more liberal.

The τ-transitions in states 1 and 2 indicate that M does not need to be able to produce tokens in p' or q' before consuming a token from both p and q. The τ-transitions in states 0 and 4 indicate that M should not depend on the order in which tokens are produced in p and q, or consumed from p' and q'.

The fact transitions check a condition related to M modeling a pair of services, viz., whether M cannot produce a token in place p' too early (before consuming a token from place p); and similarly for q and q'.

Net T: The condition on N in Section 3 is that N is sound under the restrictions from the may/exit transition system in Fig. 4, and that all transitions that are connected to p and q occur in Fig. 4. Net T in Fig. 6(b) is motivated by the first part, and therefore consists of two sides: the left side contains both the may and the exit transitions, while the right side contains only exit transitions. The initial state is at the left, the final state is at the right, and there are only τ-edges from left to right. This motivates that net T checks that N is sound under the restrictions from Fig. 4.

What remains is to ensure that N does not contain transitions that do not occur in Fig. 4. The fact transitions in T check that the transitions p and q (that

produce a token in a place) do not occur as specified in Fig. 4. We do not need such a check for transitions p' and q' (that consume a token from a place), as in these cases the place is guaranteed to be empty.

5 Generalizing Theory

In this section we focus on the approach from Section 4. We give a theoretical foundation, and show a way in which valid pairs of nets can be constructed. In this way we generalize the notion of synchronizable pairs of places.

5.1 Foundation

Suppose two open nets N and M are given, and we want to prove that their composition is sound, denoted by $SD.(N \oplus M)$. We aim for a sufficient condition that can be split into parts referring only to N or M, but not both.

Let us calculate for any two open nets N and M:

$$
\begin{array}{ll}
& SD.(N \oplus M) \\
\equiv & \{\ \text{“}\Leftarrow\text{”}\ \text{Definition 6 (Accordance pre-order); “}\Rightarrow\text{”}\ \text{instantiation}\ E := N\ \} \\
& (\exists E ::\ N \leq E\ \wedge\ SD.(E \oplus M)) \\
\equiv & \{\ \text{Proposition 1 (Deciding accordance using maximal controller)}\ \} \\
& (\exists E ::\ SD.(N \oplus mc.E)\ \wedge\ SD.(E \oplus M))
\end{array}
$$

Note that conjunct $SD.(E \oplus M)$ guarantees that net E is controllable, and hence $mc.E$ is defined. For every open net E we thus obtain a sufficient condition for proving $SD.(N \oplus M)$, viz.,

$$(\forall E, M, N ::\ SD.(N \oplus mc.E)\ \wedge\ SD.(E \oplus M)\ \ \Rightarrow\ \ SD.(N \oplus M))$$

Alternatively, suppose the open nets E and M are given. We want to prove that $SD.(E \oplus M)$ is an exact condition for concluding that for each net N such that $SD.(N \oplus mc.E)$ holds, also $SD.(N \oplus M)$ holds.

Theorem 1.

$$
\begin{array}{l}
(\forall E, M ::\ SD.(E \oplus M)\ \equiv\ (\forall N ::\ SD.(N \oplus mc.E)\ \Rightarrow\ SD.(N \oplus M))) \\
(\forall E, N ::\ SD.(N \oplus mc.E)\ \equiv\ (\forall M ::\ SD.(E \oplus M)\ \Rightarrow\ SD.(N \oplus M)))
\end{array}
$$

Proof. We justify the two equivalences $\equiv$ by proving two implications:

$\Rightarrow$: follows from our introductory result (using quantifier logic);
$\Leftarrow$: follows from instantiations $N := E$ and $M := mc.E$ respectively. □

In fact, “$\Leftarrow$” uses that $mc.E$ is a controller, and “$\Rightarrow$” uses that it is maximal.

5.2 Computing Maximal Controllers

Maximal controllers are related to canonical duals [7] for which there is a trivial computation, but this does not apply in our setting. In [19] we have shown how to construct a maximal controller if the behavioral property is deadlock freedom.

This was based on operating guidelines [16] for deadlock freedom. As far as we know, there are no published results yet wrt. soundness for maximal controllers nor for operating guidelines. In what follows we compute a finite representation of a maximal controller for soundness, but only for a class of open nets.

The open nets that we consider are a generalization of net E in Fig. 6(a) or net T in Fig. 6(b). For describing the parameters, we focus on such a net E. The states contain a set I of core states (which are numbered in Fig. 6(a)), including the initial state b. There is a subset F of I that contains the core states with a τ transition to a final state without outgoing transitions.

The transitions from each core state $i: \ i \in I$ can be characterized by three sets: set $E.i$ contains the non-τ actions from state i, set $C.i$ contains the non-τ actions from state i for which there is a dedicated τ transition from i, and set $D.i$ contains the fact non-τ actions from state i that lead to a deadlock. Finally, $W.i.a$ indicates the next core state after doing action a in core state i;

Definition 8 (Pair of nets $\langle X, Y \rangle$). *Given a set I of core states, an initial core state $b: \ b \in I$, and a set $F: \ F \subseteq I$ of final core states. Furthermore, for every $i: \ i \in I$, three sets $E.i$, $C.i$ and $D.i$ of actions are given, and a successor function $W.i.a$ with result type I for $a: \ a \in E.i \cup C.i$.*

A pair of nets $\langle X, Y \rangle$ consists of two open nets X and Y. Open net X has the following LTS semantics (for any $\delta: \ \delta \notin I$ and $\omega: \ \omega \notin I$):

$$
\begin{array}{lcl}
S & = & \{X.i \mid i \in I\} \ \cup \ \{X.i.a \mid i \in I \ \wedge \ a \in C.i\} \ \cup \ \{X.\delta, X.\omega\} \\
\rightarrow & = & \{(X.i, a, X.(W.i.a)) \mid a \in E.i\} \\
 & & \quad \cup \ \{(X.i, \tau, X.i.a) \mid a \in C.i\} \ \cup \ \{(X.i.a, a, X.(W.i.a)) \mid a \in C.i\} \\
 & & \quad \cup \ \{(X.i, a, X.\delta) \mid a \in D.i\} \ \cup \ \{(X.i, \tau, X.\omega) \mid i \in F\} \\
\Omega & = & \{X.\omega\} \qquad s_i \ = \ X.b
\end{array}
$$

Open net Y has the following LTS semantics (for any $\delta: \ \delta \notin I$):

$$
\begin{array}{lcl}
S & = & \{Y.i \mid i \in I\} \ \cup \ \{Z.i \mid i \in I\} \ \cup \ \{Y.\delta\} \\
\rightarrow & = & \{(Y.i, a, Y.(W.i.a)) \mid a \in E.i \cup C.i\} \ \cup \ \{(Z.i, a, Z.(W.i.a)) \mid a \in C.i\} \\
 & & \quad \cup \ \{(Y.i, a, Y.\delta) \mid a \in \overline{E.i \cup C.i \cup D.i}\} \ \cup \ \{(Y.i, \tau, Z.i) \mid i \in I\} \\
\Omega & = & \{Z.i \mid i \in F\} \qquad s_i \ = \ Y.b
\end{array}
$$

This definition provides a procedure for computing Y if X is given, or X if Y is given. Net E in Fig. 6(a) and net T in Fig. 6(b) form a pair of nets $\langle E, T \rangle$ as described in Definition 8.

In the remainder of this section we provide a condition on pairs of nets $\langle X, Y \rangle$ that guarantees that nets X and Y are a maximal controller of each other. To this end we need to prove that net Y is a controller of net X, and net Y is larger than each controller of X; and vice versa.

The first requirement boils down to checking soundness of $X \oplus Y$. We focus on the last requirement, which can be formalized as

- $(\forall M :: \ SD.(X \oplus M) \ \Rightarrow \ M \leq Y)$, and
- $(\forall N :: \ SD.(N \oplus Y) \ \Rightarrow \ N \leq X)$,

which are equivalent to the following symmetric formalization:

$$(\forall M, N :: \ SD.(N \oplus Y) \ \wedge \ SD.(X \oplus M) \ \ \Rightarrow \ \ SD.(N \oplus M))$$

Before proving this in Lemma 2, we first prove a supporting lemma. For a path σ, we use $\pi_N.\sigma$ to denote the projection of σ on the steps by N; we use $s \xrightarrow{\sigma} t$ to denote that there is a path σ from state s to state t. In the context of synchronous interfaces, each state that is reachable in the composition $N \oplus M$ can be described as a pair (s_N, s_M) consisting of a state s_N in N and a state s_M in M.

Lemma 1. *Given any pair of nets $\langle X, Y\rangle$ as in Definition 8, and any two open nets M and N, such that $SD.(N \oplus Y)$ and $SD.(X \oplus M)$. For any path σ and states s_N and s_M such that $N \oplus M \xrightarrow{\sigma} (s_N, s_M)$, there exists a core state i and paths σ_1 and σ_2 such that:*

$$N \oplus Y \xrightarrow{\sigma_1} (s_N, Y.i) \wedge X \oplus M \xrightarrow{\sigma_2} (X.i, s_M) \wedge \pi_N.\sigma = \pi_N.\sigma_1 \wedge \pi_M.\sigma_2 = \pi_M.\sigma$$

Proof. We prove this using structural induction on σ. In the basis $\sigma = \epsilon$ we choose $\sigma_1 = \sigma_2 = \epsilon$ and $i = b$. Each appended internal step of either N or M in σ can be appended to σ_1 or σ_2 respectively without affecting i.

Each appended synchronized step a in σ is a step of both M and N, and hence it should be appended to both σ_1 and σ_2. As $SD.(N \oplus Y)$ and $SD.(X \oplus M)$, a is not a fact transition in state $X.i$ nor in state $Y.i$, i.e., a is not in $D.i$ nor in $\overline{E.i \ \cup \ C.i \ \cup \ D.i}$; hence a is in $E.i$ or $C.i$. Both $X.i$ and $Y.i$ can perform this step (after inserting a τ-step in σ_2 in case $a \in C.i$), and resulting in state $X.(W.i.a)$ and $Y.(W.i.a)$ respectively. □

Lemma 2. *For every pair of nets $\langle X, Y\rangle$ as in Definition 8:*

$$(\forall M, N :: \ SD.(N \oplus Y) \ \wedge \ SD.(X \oplus M) \ \ \Rightarrow \ \ SD.(N \oplus M))$$

Proof. Assume $SD.(N \oplus Y)$ and $SD.(X \oplus M)$, and let us focus on $SD.(N \oplus M)$. Soundness denotes that after every path there is a path to a final state. From Lemma 1 we can conclude that every path in $N \oplus M$ to any state (s_N, s_M) can be mimicked using paths to $(s_N, Y.i)$ and $(X.i, s_M)$ in $N \oplus Y$ and $X \oplus M$.

What remains is to construct a path from state (s_N, s_M) to a final state. At this point we use the τ-step from $Y.i$ to $Z.i$. As $N \oplus Y$ is sound, there is a path from $(s_N, Z.i)$ to a final state $(s'_N, Z.i')$, with $i' \in F$; in Y such a path can only use C-actions.

As $X \oplus M$ is sound (and hence deadlock free), we can use the τ-steps to the C-actions in X to construct a path in $X \oplus M$ to a state $(X.i', s'_M)$ using the same synchronized steps as the path in $N \oplus Y$. From state $(X.i', s'_M)$ the state $(X.\omega, s'_M)$ can be reached using a τ-step. As $X \oplus M$ is sound, this means that from state s'_M in M there is a path to a final state s''_M using only internal steps. Using a proper synchronization of these paths, we obtain a path in $M \oplus N$ to a final state (s'_N, s''_M). □

Theorem 2. *Given a pair of nets $\langle X, Y\rangle$ as in Definition 8. If the composition $X \oplus Y$ is sound, then nets X and Y are a maximal controller of each other.*

The composition $T \oplus E$ for the pair $\langle T, E \rangle$ in Fig. 6 is sound. Given the disjointness of C, D, E and $\overline{E.i \cup C.i \cup D.i}$, and as the C's are non-empty, there are no reachable deadlocks. From every reachable state there is a path to a final state that only uses C transitions. Hence T and E are maximal controllers.

Using the construction from this section, we can generalize Section 4 from pairs of synchronizable places to larger numbers of synchronizable places. In these cases, the nets E and T become bigger; net E is probably the simplest one to modify manually, whereas net T can better be computed using Definition 8.

6 Conclusions and Further Work

In the context of service compositions, we have developed conditions on a refined net N and a refining net M in isolation such that the refinement of N by M is sound. We have generalized these techniques to a larger class of refinements, and proved their correctness in terms of composition and maximal controllers.

It is further work to combine our techniques (aimed at vertical modularization) with the techniques from [2] for horizontal modularization. Furthermore, we want to investigate which other temporal properties, besides soundness, can be preserved by (extensions of) our method.

Our techniques are based on pairs of nets that are each others maximal controller. These nets can be considered as tests, and thus have a much wider application area. For example, a sound pair $\langle X, Y \rangle$ can be seen as a contract between two organizations. If one organization develops a service M that is sound with X, and the other independently develops a service N that is sound with Y, then the composition of M and N is also guaranteed to be sound. In this way, X and Y can be seen as test stubs and skeletons for M and N.

References

1. van der Aalst, W.M.P.: Verification of workflow nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
2. van der Aalst, W.M.P., van Hee, K.M., Massuthe, P., Sidorova, N., van der Werf, J.M.E.M.: Compositional service trees. In: Franceschinis, G., Wolf, K. (eds.) PETRI NETS 2009. LNCS, vol. 5606, pp. 283–302. Springer, Heidelberg (2009)
3. van der Aalst, W.M.P., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: Multiparty contracts: Agreeing and implementing interorganizational processes. The Computer Journal (2009)
4. Brauer, W., Gold, R., Vogler, W.: A survey of behaviour and equivalence preserving refinements of Petri nets. In: ATPN 1990. LNCS, vol. 483, pp. 1–46 (1991)
5. Bravetti, M., Tennenholtz, M.: Contract based multi-party service composition. In: Arbab, F., Sirjani, M. (eds.) FSEN 2007. LNCS, vol. 4767, pp. 207–222. Springer, Heidelberg (2007)
6. Brinksma, E.: A theory for the derivation of tests. In: Protocol Specification, Testing, and Verification VIII, pp. 63–74. North-Holland, Amsterdam (1988)
7. Castagna, G., Dezani-Ciancaglini, M., Giachino, E., Padovani, L.: Foundations of session types. In: PPDP 2009, pp. 219–230. ACM, New York (2009)

8. Chow, Y.S., Robbins, H., Siegmund, D.: The Theory of Optimal Stopping: Great Expectations. Houghton Mifflin Company (1971)
9. Clarke, E., Emerson, E.: Design and synthesis of synchronization skeletons using branching-time temporal logic. In: Logic of Programs 1981. LNCS, vol. 131, pp. 52–71 (1982)
10. Dill, D.L.: Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits. MIT Press, Cambridge (1989)
11. Genrich, H.J., Thieler-Mevissen, G.: The calculus of facts. In: MFCS 1976. LNCS, vol. 45, pp. 588–595 (1976)
12. van Hee, K.M., Sidorova, N., Voorhoeve, M.: Soundness and separability of workflow nets in the stepwise refinement approach. In: van der Aalst, W.M.P., Best, E. (eds.) ATPN 2003. LNCS, vol. 2679, pp. 337–356. Springer, Heidelberg (2003)
13. Kindler, E.: A compositional partial order semantics for Petri net components. In: ATPN 1997. LNCS, vol. 1248, pp. 235–252 (1997)
14. Kindler, E., Martens, A., Reisig, W.: Inter-operability of workflow applications: Local criteria for global soundness. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) BPM 2000. LNCS, vol. 1806, pp. 235–253. Springer, Heidelberg (2000)
15. Larsen, K.G., Thomsen, B.: A modal process logic. In: LICS 1988, pp. 203–210 (1988)
16. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: Kleijn, J., Yakovlev, A. (eds.) ATPN 2007. LNCS, vol. 4546, pp. 321–341. Springer, Heidelberg (2007)
17. Malik, R., Streader, D., Reeves, S.: Conflicts and fair testing. Journal of Foundations of Computer Science 17(4), 797–813 (2006)
18. Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. Annals of Mathematics, Computing & Teleinformatics 1(3), 35–43 (2005)
19. Mooij, A.J., Parnjai, J., Stahl, C., Voorhoeve, M.: Constructing substitutable services using operating guidelines and maximal controllers (2010) (accepted for WS-FM 2010)
20. Mooij, A.J., Stahl, C., Voorhoeve, M.: Relating fair testing and accordance for service replaceability. Journal of Logic and Algebraic Programming 79(3–5), 233–244 (2010)
21. Mooij, A.J., Voorhoeve, M.: Proof techniques for adapter generation. In: Bruni, R., Wolf, K. (eds.) WS-FM 2008. LNCS, vol. 5387, pp. 207–223. Springer, Heidelberg (2009)
22. Mooij, A.J., Voorhoeve, M.: Trading off concurrency to generate behavioral adapters. In: ACSD 2009, pp. 109–118. IEEE, Los Alamitos (2009)
23. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (1989)
24. Rensink, A., Vogler, W.: Fair testing. Information and Computation 205(2), 125–198 (2007)
25. Schmidt, K.: LoLA: A low level analyser. In: Nielsen, M., Simpson, D. (eds.) ATPN 2000. LNCS, vol. 1825, pp. 465–474. Springer, Heidelberg (2000)
26. Siegeris, J., Zimmermann, A.: Workflow model compositions preserving relaxed soundness. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 177–192. Springer, Heidelberg (2006)
27. Stahl, C., Wolf, K.: Deciding service composition and substitutability using extended operating guidelines. Data Knowl. Eng. 68(9), 819–833 (2009)
28. Suzuki, I., Murata, T.: A method for stepwise refinement and abstraction of Petri nets. Journal of Computer and System Sciences 27, 51–76 (1983)
29. Vogler, W.: Modular Construction and Partial Order Semantics of Petri Nets. LNCS, vol. 625. Springer, Heidelberg (1992)