# Checking Compatibility of Web Services Behaviorally

Kais Klai, Hanen Ochi



# Checking Compatibility of Web Services Behaviorally

Kais Klai[1] and Hanen Ochi[2]

[1] Institut TELECOM SudParis , CNRS UMR Samovar
9 rue Charles Fourier 91011 Evry, France
kais.klai@telecom-sudparis.eu
[2] LIPN, CNRS UMR 7030, Université Paris 13
99 avenue Jean-Baptiste Clément
F-93430 Villetaneuse, France
hanen.ochi@lipn.univ-paris13.fr

Web services composition is an emerging paradigm for enabling application integration within and across organizational boundaries. In this context, we propose an approach based on Symbolic Observation Graphs (SOG) allowing to decide whether two (or more) web services can cooperate safely. The compatibility between two web services is defined by the well known soundness property on open workflow nets. This property guarantees the absence of anomalies (e.g. deadlock) that can appear after composition. We propose to abstract the concrete behavior of a web service using a SOG and show how composition of web services as well as the compatibility check can be achieved through the composition of their abstractions (i.e. SOGs). This approach allows to respect the privacy of the services since SOGs are based on collaborative activities only and hide the internal structure and behavior of the corresponding service.

## 1 Introduction

Service oriented architecture (SOA) has evolved to become a promising technology for the integration of disparate software components using Internet protocols. These components, called *Web Services*, are available in the distributed environment of the Internet. Organizations attempt to provide their own services to be matched with others following a request, their complex tasks are resolved using a combination of several web services. For automatically selecting and composing services in a well-behaved manner, information about the services has to be exposed. Usually, web services are published by giving their public description behavior in a repository, such as Universal Description, Discovery and Integration UDDI, in order to make possible the collaboration with potential requesters. In particular, this information must be sufficient to decide whether the composition of two services is possible. However, organizations usually want to hide the trade secrets of their services and thus need to find a proper abstraction which is published instead of the service itself in the repository. Thus, the public abstraction should satisfy two contradictory requirements: on one hand, it should respect the privacy of the underlying organization. On the other hand, it

should supply enough information to allow the collaboration and the communication with potential partners in a correct way. Thus, correctness of the original composite web service should be detected from the analysis of the composition of the corresponding public abstractions. Among other abstraction approaches, the Symbolic Observation Graph (SOG) based technique, initially introduced for model checking of concurrent systems [4] and then applied to the verification of inter enterprise business processes [8,11], is promising. A SOG is a graph whose construction is guided by a subset of *observed* actions. The nodes of a SOG are aggregates hiding a set of local states which are connected with non observed actions. The arcs of a SOG are exclusively labeled with observed actions. Thus, we propose to use SOGs as abstraction of web services. By observing the collaborative activities of a web service, publishing a SOG as an abstraction allows to hide its internal behavior inside the aggregates. The strength of such approach is that a SOG associated with a web service represents a reduced abstraction of its reachable state space while preserving its behavioral properties (e.g. deadlock freeness, temporal properties, ...). Checking the compatibility of two web services is reduced to check the compatibility on the composition of their SOGs.

In this paper, a web service is formally represented by an oWF-net [14]. Two web services are said to be compatible if the composite oWF-net is sound [17]. The *soundness* property on a oWF-net is defined by three requirements: (1) *option to complete*: starting from any reachable state, it is possible to reach a final state, (2) *proper completion*: there is no reachable state strictly greater than a final state, and, (3) *no dead transitions*: each action is executed at least in one reachable state. Although, in practice, the behavior of web services is frequently described using industrial description languages such as BPEL4WS, BPWL and WSCI, several approaches allow to map these models to the formal description languages [13][6] (Petri nets). Thus, our approach is relevant for a very broad class of modeling languages and we can use an UDDI registry as a repository to extract web service's specifications for this purpose.

This paper is organized as follows: first, Section 2 presents some preliminary notions on oWF-nets, their composition and the notion of *soundness*. Then, a running example is presented in Section 3 allowing to illustrate our approach through the paper. In Section 4, we present symbolic observation graphs and how the soundness property is preserved by such an abstraction. Composition of SOGs and checking the compatibility property is the issue of Section 5. In Section 6, we discuss some related works. Finally, Section 7 concludes the paper and presents some aspects of the future work.

## 2 Preliminaries

### 2.1 Description models

**Petri nets** The need for formal methods and software tools for describing and analyzing web services is widely recognized. Petri nets [15], a well known formalism for modeling real-time systems, can be used for describing and analyzing the behavior of web services.

**Definition 1.** *A Petri net is 4-tuple* $N = \langle P, T, F, W \rangle$ *where:*

- $P$ *is a finite set of places (circles) and* $T$ *a finite set of transitions (squares) with* $(P \cup T) \neq \emptyset$ *and* $P \cap T = \emptyset$,
- *A flow relation* $F \subseteq (P \times T) \cup (T \times P)$,
- $W : F \to \mathbb{N}^+$ *is a mapping that assigns a positive weight to any arc.*

Each node $x \in P \cup T$ of the net has a pre-set and a post-set defined respectively as follows: ${}^\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$, and $x^\bullet = \{y \in P \cup T \mid (x, y) \in F\}$. Adjacent nodes are then denoted by ${}^\bullet x^\bullet = {}^\bullet x \cup x^\bullet$. The incidence matrix $C$ associated with the net is defined as follows : $\forall (p, t) \in P \times T : C(p, t) = W(t, p) - W(p, t)$

A marking of a Petri net $N$ is a function $m : P \to \mathbb{N}$. The initial marking of $N$ is denoted by $M_0$. The pair $(N, M_0)$ is called a Petri net system.

A transition $t$ is said to be enabled by a marking $m$ (denoted by $m \xrightarrow{t}$) iff $\forall p \in {}^\bullet t$, $W(p, t) \leq m(p)$. If a transition $t$ is enabled by a marking $m$, then its firing leads to a new marking $m'$ (denoted by $m \xrightarrow{t} m'$) s.t. $\forall p \in P : m'(p) = m(p) + C(p, t)$. Given a set of markings $S$, we denote by $Enable(S)$ the set of transitions enabled by elements of $S$. The set of markings reachable from a marking $m$ in $N$ is denoted by $R(N, m)$. The set of markings reachable from a marking $m$, by firing transitions of a subset $T'$ only is denoted by $Sat(m, T')$. By extension, given a set of markings $S$ and a set of transitions $T'$, $Sat(S, T') = \bigcup_{m \in S} Sat(m, T')$ . For a marking $m$, $m \not\to$ denotes that $m$ is a dead marking, i.e., $Enable(\{m\}) = \emptyset$.

**oWF-nets** We define a web service by its behavior and its interface. An instance of a given service corresponds to an execution of this service. The interface consists of a set of ports. A pair of ports can be connected using a channel, thus enabling the exchange of messages sent or received by services. A web service can be viewed as a control structure describing its behavior according to an interface to communicate asynchronously with other services in order to reach a final state (i.e. a state representing a proper termination). We use a particular Petri net for modeling the control-flow dimension of a web service, called *open Work-Flow net (oWF-net)* and introduced in [14]. It is essentially a liberal version of workflow nets [1], enriched with communication places representing the interface. Each communication place models a channel to send (receive) messages to (from) another oWF-net. Transitions in a oWF-net correspond to activities and places represent pre-conditions for activities.

**Definition 2.** *An open workflow net (oWF-net for short) is defined by a tuple* $N = \langle P, T, F, W, m_0, I, O, \Omega \rangle$ *where:*

- $\langle P \cup I \cup O, T, F, W \rangle$ *is a Petri net;*
- $m_0$ *is the initial marking;*
- $I$ *(resp.* $O$*) is a set of input (resp. output) places (*$I \cup O$ *represents the set of interface places) satisfying:*

- $(I \cup O) \cap P = \emptyset$
- $\forall p \in I : {}^\bullet p = \emptyset$ *(input interfaces places)*
- $\forall p \in O : p^\bullet = \emptyset$ *(output interface places)*

– $\Omega$ *is a set of final markings.*

From now on, given an oWF-net $N$, the subnet $N^* = \langle P, T, F^*, W^* \rangle$ is called the *inner net* of $N$. $F^*$ and $W^*$ are derived (by projection) from $F$ and $W$, respectively, by removing the input and the output interface places of $N$.

Based on the notion of oWF-nets, we have to analyze the behavior of web services from the local point of view. So we can check the soundness property [17] to detect the anomalies on web services. The *soundness* property on a oWF-net $N$ concerns its inner Petri net $N^*$ and is defined by three requirements: (1) *option to complete*: starting from any reachable marking, it is possible to reach a final marking, (2) *proper completion*: there is no reachable marking strictly greater than a final marking, and, (3) *no dead transitions*: each transition is firable at least in one reachable marking.

**Definition 3.** *Let $N = \langle P, T, F, W, m_0, I, O, \Omega \rangle$ be an oWF-net. $N$ is sound iff the following requirements are satisfied:*

– *option to complete:* $\forall m \in R(N^*, m_0),\ \exists m_f \in \Omega$ *s.t.* $m_f \in R(N^*, m)$*;*
– *proper completion:* $\forall m \in R(N^*, m_0),\ \forall m_f \in \Omega\ m \geq m_f \implies m = m_f$*;*
– *no dead transitions:* $\forall t \in T,\ \exists m \in R(N^*, m_0)$ *s.t.* $m \xrightarrow{t}$.

**Composition of web services** The basic web services infrastructure provides simple interactions between a client and a web service. However, the implementation of a web service's business needs generally the invocation of other web services. Thus it is necessary to combine the functionalities of several web services. The process of developing a composite service is called service composition.

Composite services are recursively defined as an aggregation of elementary and composite services. The composition of two or more services generates a new service providing both the original behavior of initial services and a new collaborative behavior for carrying out a new composite task. From modeling point of view, a composite service can be described as a recursive composition of oWF-nets. Communication between services takes place by exchanging messages via interface places. Thus, composing two oWF-nets is modeled by merging their respective shared constituents which are the equally labeled input and output interface places. Such a fused interface place models a channel and a token on such a place corresponds to a pending message in the respective channel. As it is convenient to require that all communications are bilateral and directed, i.e., every interface place $p \in (I \cup O)$ has only one oWF-net that sends into $p$ and only one oWF-net that receives from $p$. Thereby, oWF-nets involved in a composition are pairwise *interface compatible*.

**Definition 4.** *Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two oWF-nets with pairwise disjoint constituents except for interfaces. If only input places of one oWF-net overlap with output places of the other oWF-net, i.e., $I_1 \cap I_2 = \emptyset$ and $O_1 \cap O_2 = \emptyset$, then $\mathcal{N}_1$ and $\mathcal{N}_2$ are* interface compatible.

**Definition 5.** *Let $\mathcal{N}_i = \langle P_i, T_i, F_i, W_i, m_{0i}, I_i, O_i, \Omega_i \rangle$, for $i \in \{1, 2\}$, be two interface compatible oWF-nets. Their composition, namely $\mathcal{N}_1 \oplus \mathcal{N}_2$, is the oWF-net $N = \langle P, T, F, W, m_0, I, O, \Omega \rangle$ defined as follows:*

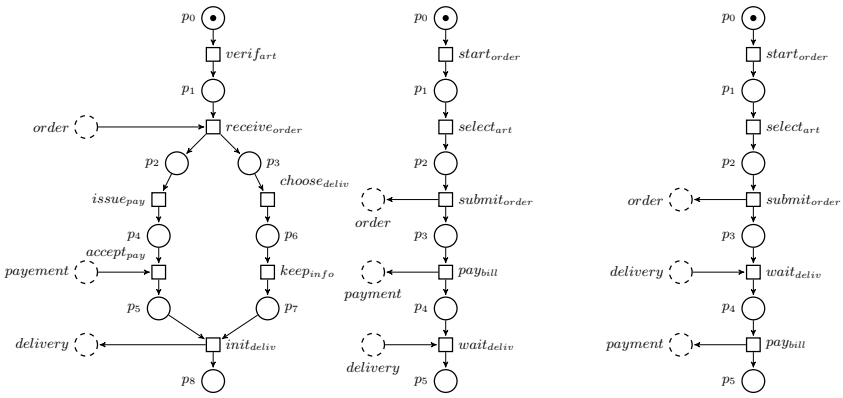
- $P = P_1 \cup P_2$, $T = T_1 \cup T_2$, $F = F_1 \cup F_2$, $W = W_1 \oplus W_2$
- $I = (I_1 \cup I_2) \setminus (O_1 \cup O_2)$, $O = (O_1 \cup O_2) \setminus (I_1 \cup I_2)$,
- $m_0 = m_{01} \oplus m_{02}$ *and* $\Omega = \Omega_1 \oplus \Omega_2$.

The oWF-net composition is commutative and associative i.e. for interface compatible oWF-nets $\mathcal{N}_1$, $\mathcal{N}_2$ and $N_3$: $\mathcal{N}_1 \oplus \mathcal{N}_2 = \mathcal{N}_2 \oplus \mathcal{N}_1$ and $(\mathcal{N}_1 \oplus \mathcal{N}_2) \oplus N_3 = \mathcal{N}_1 \oplus (\mathcal{N}_2 \oplus N_3)$. An oWF-net with an empty interface ($I = \emptyset$ and $O = \emptyset$) is called a *closed net*.

A composite web service modeled as a closed net is a service that consists of the coordination of several conceptually autonomous but interface compatible services (open nets). Although, it is not easy to specify how this coordination should behave, we focus here on semantic compatibility between web services.

**Definition 6.** *Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two interface compatible open nets and let $N = \mathcal{N}_1 \oplus \mathcal{N}_2$. Then, $\mathcal{N}_1$ is said to be **compatible** with $\mathcal{N}_2$ iff $N$ is sound.*

## 3 Running Example




(a) Online shop (b) Customer $C_1$ (c) Customer $C_2$

**Fig. 1.** The oWF-nets of an online shop and two customers

Throughout this paper, we use an example of three web services, inspired from [16] (see Figure 1): an online shop and two different customers. The example is modeled using oWF-nets. The dashed circles denote the interface places

(input/output places). While browsing an online shop, the first customer (Figure 1(b)) selects items he is interested in, pays his bill and proceeds for delivery step. For the online shop (Figure 1(a)), once the order is submitted, the subsequent payment handling and the verification process of delivery are triggered. These two tasks can be done concurrently. After verifying information about payment, the order is automatically delivered. Figure 1(c) represents a customer who behaves in a different way, since he pays his bill only after receiving the goods already bought. Note that all these oWF-nets are sound locally, and that the online shop's model is interface compatible with both customers' models.

## 4 Symbolic Observation Graph

### 4.1 Abstraction of web services

In this section we propose to use symbolic observation graphs (SOG) [4,12] in order to abstract oWF-nets. Exposing a SOG related to a service allows to hide internal activities while checking compatibility is still possible using locally computed information. Before we give the definition of a SOG, let us give some basic notations.

**Observed actions** Given an oWF-net $N$, we distinguish the transitions connected to the interface places, called *interface transitions*, from the internal transitions. The first are called *observed transitions* while the last are called *unobserved transitions*.

**Definition 7.** *Let $N = \langle P, T, F, W, m_0, I, O, \Omega \rangle$ be an oWF-net. The sets of observed transitions (Obs) and unobserved transitions (UnObs) are respectively defined as follows:*

- $Obs = \{t \in T \mid ({}^\bullet t \cup t^\bullet) \cap (I \cup O) \neq \emptyset\}$,
- $UnObs = T \setminus Obs$.

**Observed behavior** Given an oWF-net $N$, the *observed behavior* is defined as a mapping applied on the reachable markings, $R(N^*, m_0)$, of the inner net $N^*$. It is then extended progressively to sets of states. It will be established that the *observed behavior* is the necessary and sufficient local information to be retained so that compatibility between two web services can be checked. For this purpose, and for the remaining part of this paper, we assume an additional *virtual* observed transition **term** belonging to $Obs$. Observing **term** means that the system properly terminates. In the following, we denote by $Sat(S)$ the set of markings reachable from a marking $m \in S$, by firing only unobserved transitions (i.e., $Sat(S, UnObs)$).

**Definition 8.** *Let $N = \langle P, T, F, W, m_0, I, O, \Omega \rangle$ be an oWF-net. The observed behavior is progressively defined by :*

1. $\lambda_{\mathcal{N}} : R(N^*, m_0) \rightarrow 2^{Obs}$
   $\lambda_{\mathcal{N}}(m) = \begin{cases} \bullet\ (Enable(Sat(m)) \cap Obs) \cup \{term\} & \text{if } Sat(m) \cap \Omega \neq \emptyset \\ \bullet\ Enable(Sat(m)) \cap Obs & \text{otherwise} \end{cases}$
2. $\lambda_{\mathcal{N}} : 2^{(R(N^*, m_0))} \rightarrow 2^{2^{Obs}}$
   $\lambda_{\mathcal{N}}(S) = \{\lambda_{\mathcal{N}}(m) \mid m \in S\}$
3. $\lambda_{min} : 2^{R(N^*, m_0)} \rightarrow 2^{2^{Obs}}$
   $\lambda_{min}(S) = \{X \in \lambda_{\mathcal{N}}(S) \mid \nexists Y \in \lambda_{\mathcal{N}}(S) :\ Y \subset (X \setminus \{term\})\}$

Informally, for each marking $m$ in $R(N^*, m_0)$, the observed behavior of $m$, $\lambda_{\mathcal{N}}(m)$, represents the set of observed actions which can be executed from $m$, possibly via a sequence of unobserved actions. In addition, *term* is a member of $\lambda_{\mathcal{N}}(m)$ if and only if a final marking is reachable from $m$ using unobserved actions only. The observed behavior $\lambda_{\mathcal{N}}$ associated with a set of markings $S$ is a set of sets of observed actions. This set contains the observed behavior of the markings of $S$. Finally, the observed behavior mapping $\lambda_{min}$ applied to a set of markings $S$ is the minimal set of subsets (w.r.t. the set inclusion relation) of $\lambda_{\mathcal{N}}(S)$. The inclusion relation does not concern the *term* action. For instance, if there exist two markings $m, m' \in S$ such that $\lambda_{\mathcal{N}}(m) = \emptyset$ and $\lambda_{\mathcal{N}}(m') = \{term\}$, then both sets $\emptyset$ and $\{term\}$ will belong to $\lambda_{min}(S)$. This way we distinguish a dead marking from a final marking reached in $S$.

From now on, a state (marking) $m$ is said to be dead if and only if its observed behavior is the empty set. This generalizes the original definition of a dead state since a terminal livelock (a livelock from which no observed action is enabled) is considered as a deadlock as well.

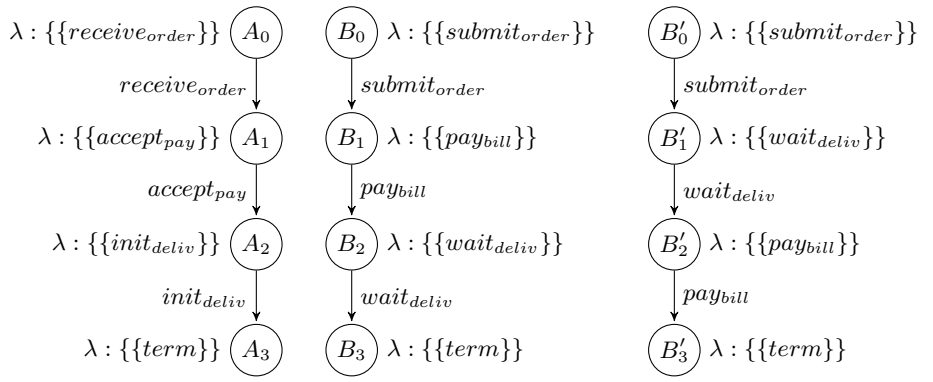
**Symbolic Observation Graph** The construction of the *SOG* corresponding to an oWF-net is guided by the set of observed transitions. A SOG is defined as a graph where each node is a set of markings linked by unobserved transitions and each arc is labeled by an observed transition. Nodes of the *SOG* are called *aggregates* and may be represented and managed efficiently using decision diagram techniques (BDDs, see e.g., [2]). In practice, due to the small number of observed transitions in loosely coupled oWF-nets, the SOG has a very moderate size and thus the time complexity of the verification process is negligible in comparison to the building time of the SOG (see [4,10,9,8] for experimental results). Before we define the SOG, let us define what an aggregate is.

**Definition 9.** *Let $N = \langle P, T = Obs \cup UnObs, F, W, m_0, I, O, \Omega \rangle$ be an oWF-net. An* aggregate *of $N$ is a couple $a = \langle S, \lambda \rangle$ defined as follows:*

1. *$S$ is a nonempty subset of $R(N^*, m_0)$ s.t.: $m \in S \Leftrightarrow Sat(m) \subseteq S$;*
2. *$\lambda = \lambda_{min}(S)$.*

From now on, $a.S$ and $a.\lambda$ denote the attributes of a given aggregate $a$. Note that the observed behavior attached to an aggregate allows to determine whether it is a final aggregate or not and whether it contains a deadlock or not. Indeed, an aggregate $a$ contains a dead marking iff $\emptyset \in a.\lambda$. It contains a final marking iff

$\exists Q \in a.\lambda$: $term \in Q$. In practice, since an aggregate is represented by a BDD, the computation of the corresponding observed behavior should be performed symbolically (using sets operations). A symbolic algorithm for the computation of the observed behavior is proposed in [7].




(a) A SOG of online shop (b) A SOG of customer $C_1$ (c) A SOG of customer $C_2$

**Fig. 2.** SOGs of the running example models

**Definition 10.** *A* symbolic observation graph *(SOG($\mathcal{N}$) for short) is a 5-tuple* $\langle \mathcal{A}, Act, \rightarrow, a_0, \Omega' \rangle$ *associated with an oWF-net* $N = \langle P, T, F, W, m_0, I, O, \Omega \rangle$, *s.t.* $T = Obs \cup UnObs$, *where:*

1. $\mathcal{A}$ *is a finite set of aggregates satisfying:*
   - *If for some* $a \in \mathcal{A}$ *and* $t \in Obs$ *the set* $Ext(a,t) = \{m' \notin a.S \mid \exists m \in a.S, m \xrightarrow{t} m'\}$ *is not empty, then there exist non-empty pairwise disjoint sets* $S_1 \dots S_k$ *s.t.* $Ext(a,t) = S_1 \cup \dots S_k$, *and* $\forall i = 1 \dots k$, *there exists an aggregate* $a_i \in \mathcal{A}$ *s.t.* $a_i.S = Sat(S_i, UnObs)$.
2. $Act = Obs$;
3. $\rightarrow \subseteq \mathcal{A} \times Act \times \mathcal{A}$ *is the transition relation satisfying:*
   - *if* $a \neq a'$, $(a,t,a') \in \rightarrow$ *iff* $Ext(a,t) \neq \emptyset$ *and* $a'.S = Sat(S', UnObs)$ *for some* $S' \subseteq Ext(a,t)$.
   - $(a,t,a) \in \rightarrow$ *iff* $Sat(\{m' \in R(N^*, m_0) \mid \exists m \in a.S, m \xrightarrow{t} m'\}, UnObs) = a.S$
4. $a_0$ *is the initial aggregate s.t.* $a_0.S = Sat(m_0, UnObs)$.
5. $\Omega'$ *is a set of final aggregates defined by* $\Omega' = \{a \in \mathcal{A} \mid a.S \cap \Omega \neq \emptyset\}$.

Notice that Definition 10 does not guarantee the uniqueness of a SOG for a given open net. In fact, it supplies a certain flexibility for its implementation. In particular, the SOG can be nondeterministic. It is clear that the canonical minimal SOG is obtained when the SOG is deterministic. However, one can take advantage of the nondeterminism to obtain smaller aggregates. Indeed, when

two (for instance) states within an aggregate $a$ enable an observed transition $t_o$, then $a$ has one successor $a'$ if the SOG is deterministic and two successors $a'_1$ and $a'_2$ (s.t. $a'_1 \cup a'_2 = a'$) if not. Thus, even if the SOG obtained by this way has more aggregates, its construction might consume less time and memory (aggregate's size is smaller). Our definition generalizes the one given in [12]. The construction algorithm given in [4] is an implementation where the obtained graph is deterministic.

Figure 2 shows the SOGs associated with the oWF-nets of Figure 1. Figure 2(a) shows the SOG of the online shop model while Figure 2(b) (respectively Figure 2(c)) illustrates the SOG of the customer model $C_1$ (respectively $C_2$). Each aggregate is annotated with the corresponding observed behavior and one can see that all these SOGs are sound. Moreover, the SOG of the online shop is the one which most abstract the behaviors of the original model since it has more local behaviors. Indeed, its reachability graph contains 12 reachable markings and 15 arcs against 4 aggregates and 3 arcs in the corresponding SOG.

In the following, we establish that the soundness of a oWF-net can be checked by analyzing the corresponding SOG. As for a marking $m$, the set of aggregates reachable from a given aggregate $a$ is denoted by $R(a)$.

**Theorem 1.** *Let $\mathcal{G} = \langle \mathcal{A}, Act, \rightarrow, a_0, \Omega' \rangle$ be a SOG associated with an oWF-net $N$. $N$ is sound iff the following requirements are satisfied:*

- *option to complete:* $\forall a \in \mathcal{A}$, $\emptyset \notin a.\lambda \wedge \exists a_f \in \Omega' \mid a_f \in R(a)$.
- *proper completion:* $\forall a \in \mathcal{A}$, $\forall m \in a.S$, $\forall m_f \in \Omega$, $m \geq m_f \implies m = m_f$;
- *no dead transitions :* $\bigcup_{a \in \mathcal{A}} Enable(a.S) = T$

From the local point of view, the internal behaviors of a service are available. Thus, states inside aggregates can be analysed to check the soundness requirements but this should be done symbolically so that the efficiency of the BDD-based representation and management of the aggregates is preserved.

## 5 Synchronized product of SOGs

### 5.1 Composition of SOGs

In this section, we tackle the main idea of this paper: we will define how we compose two (ore more) web services (each ignoring internal details about the other). Starting from two interface compatible oWF-nets $\mathcal{N}_1$ and $\mathcal{N}_2$ which are already locally sound, this section shows how to check their compatibility using their respective SOGs $\mathcal{G}_1$ and $\mathcal{G}_2$. Our objective is to reduce the verification of the compatibility between $\mathcal{N}_1$ and $\mathcal{N}_2$ (structure of $\mathcal{N}_1 \oplus \mathcal{N}_2$ is unavailable anyway) to the analysis of the composition of $\mathcal{G}_1$ and $\mathcal{G}_2$, namely $\mathcal{G}_1 \oplus \mathcal{G}_2$. To reach this goal, and in order to take into account the asynchronous composition between $\mathcal{N}_1$ and $\mathcal{N}_2$, we assume that each oWF-net exposes its input and output places (resp. transitions). Then we define a medium net $N_{12}$ as an open net representing the interface between $\mathcal{N}_1$ and $\mathcal{N}_2$.

**Definition 11.** *Let $N_i = \langle P_i, T_i, F_i, W_i, m_{0i}, I_i, O_i, \Omega_i \rangle$, for $i = 1, 2$, be two interface compatible oWF-nets. The medium net related to $\mathcal{N}_1$ and $\mathcal{N}_2$, denoted by $N_{12} = \langle P_{12}, T_{12}, F_{12}, F_{12}, m_{012}, \Omega_{12} \rangle$, is the closed net defined as follows :*

- $P_{12} = (I_1 \cap O_2) \cup (O_1 \cap I_2)$
- $T_{12} = \{t \in T_i;\ {}^\bullet t^\bullet \cap ((I_i \cap O_j) \cup (O_i \cap I_j)) \neq \emptyset\}$ *for* $i, j \in \{1, 2\}$ *and* $i \neq j$
- $F_{12} = F_1|_{(P_{12} \times T_{12}) \cup (T_{12} \times P_{12})} \cup F_2|_{(P_{12} \times T_{12}) \cup (T_{12} \times P_{12})}$
- $W_{12} = W_1|_{F_{12}} \cup W_2|_{F_{12}}$
- $m_{012} = \{0\}$ *i.e. all places are empty*
- $\Omega_{12} = \{m_{012}\}$

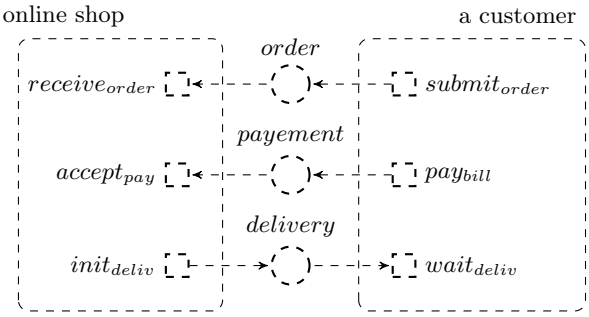



**Fig. 3.** The medium net of the running example

The transitions of the medium net are the interface transitions of $\mathcal{N}_1$ and $\mathcal{N}_2$ while its places are their interface places.

It is clear that the set of reachable markings of the medium net is infinite. However, if we assume that the composed net $\mathcal{N}_1 \oplus \mathcal{N}_2$ is bounded, then the number of states that are reachable by the interface places is finite. If the bound of an interface place is $n$ then this place can be in $n+1$ different states at most. Under such an assumption and knowing the bound of each place of the medium net, one can build a reachability graph that covers all the possible behaviors related to the interface places in $\mathcal{N}_1 \oplus \mathcal{N}_2$. The obtained graph is called *interface graph* and is defined as the following:

**Definition 12.** *Consider two oWF-nets $\mathcal{N}_1$ and $\mathcal{N}_2$ and their medium net $N_{12}$. For each place $p_i$ (for $i = 1 \ldots m$) of $N_{12}$, let $n_i$ be the bound of $p_i$ in $\mathcal{N}_1 \oplus \mathcal{N}_2$. For sake of simplicity, assume that each place $p_i$ has a single input transition $in_i$ and a single output transition $out_i$. Then, the interface graph is a a tuple $\langle \Gamma, Act, \rightarrow, m_0, \Omega \rangle$ s.t.:*

1. $\Gamma = \{\langle x_1, \ldots x_m \rangle \mid 0 \leq x_1 \leq n_i \ldots 0 \leq x_m \leq n_m\}$
2. $Act = \{in_i \mid i = 1, \ldots, m\} \cup \{out_i \mid i = 1, \ldots, m\}$
3. $\rightarrow \subseteq \Gamma \times Act \times \Gamma$ *is a transition relation such that:*

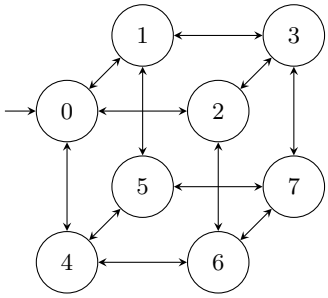



**Fig. 4.** Interface graph of the medium net

*(a)* $m \xrightarrow{in_i} m'$ *iff* $m'(b_i) = m(b_i) + 1 \wedge m'(b_i) \leq n_i$
*(b)* $m \xrightarrow{out_i} m'$ *iff* $m'(b_i) = m(b_i) - 1 \wedge m'(b_i) \geq 0$

4. $m_0 = \langle 0, \ldots, 0 \rangle$ *is the initial marking*
5. $\Omega = \{m_0\}$ *is the set of final markings*

The above definition constructs a reachability graph where each marking represents a possible configuration of the interface places of $\mathcal{N}_1$ and $\mathcal{N}_2$. The transition relation allows the evolution of the interface places' states in the following manner: a successor of a given marking is a marking where the number of tokens in one interface place has been increased or decreased (by one). Moreover, the initial marking (which is the final marking as well) is such that all the interface places are empty.

By observing all the transitions of the medium net, the interface graph of the medium net can be seen as a SOG. In this SOG, the aggregates are singletons (each reachable marking is an aggregate) and the observed behavior of each aggregate is also a singleton : the set of transitions appearing on the outgoing arcs of the corresponding marking. Finally, the set of final aggregates is again a singleton containing the initial aggregate.

Figure 4 illustrates the SOG associated with the medium net of Figure 3. The binary representation of each state number gives the state of the interface places (order, payment and delivery respectively). For instance, state number 5 stands for 101, i.e., only the interface place of payment is not marked. Unlike the SOGs associated with $\mathcal{N}_1$ and $\mathcal{N}_2$, the SOG of the medium net is not supposed to be built a priori. Thus, the bounds of the places of $N_{12}$ are not supposed to be known, as long as the composed net $\mathcal{N}_1 \oplus \mathcal{N}_2$ is bound. In the following, the SOG of the medium will be computed on-the-fly during the composition of $\mathcal{G}_1$ and $\mathcal{G}_2$. The composition of $\mathcal{G}_1$ and $\mathcal{G}_2$, denoted by $\mathcal{G}_1 \oplus \mathcal{G}_2$ is then defined as a synchronized product between three SOGs corresponding to $\mathcal{N}_1$, $N_{12}$ and $\mathcal{N}_2$ respectively. Before we define the composition of SOGs, it is important to first show how, using observed behavior of three aggregates $a_1$, $a_2$ and $a_{12}$ of $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_{12}$ respectively, one can compute the observed behavior of the aggregate resulting from their composition.

Note that the set of states $a.S$ of an aggregate $a$ has not to be stored explicitly within an aggregate. Once the SOG is built, it will not play any role in the composition process. However, since our goal is to reduce the compatibility check of two oWF-nets to the analyzing of their SOGs, we need to know which are the enabled transitions (especially local transitions) in each aggregate. Given a oWF-net $N = \langle P, T, F, W, m_0, I, O, \Omega \rangle$ and an associated SOG $G$ with respect to the set of observed transitions $Obs$, an aggregate of $G$ is henceforth identified by its observed behavior $\lambda$ and the set of enabled local transitions, namely $E$. Formally, $a.E = \{t \in T \setminus Obs \mid \exists m \in a.S,\ m \xrightarrow{t}\}$.

**Definition 13.** *Let $\mathcal{G}_i$, for $i = 1, 2$, be two SOGs associated with two oWF-nets and let $\mathcal{G}_{12}$ be the SOG associated with their medium net. Let $a_1$, $a_2$ and $a_{12}$ be three aggregates of these SOGs respectively. The* product aggregate $a = (a_1, a_{12}, a_2)$ *is defined by:*

1. $a.\lambda = \{((x \cap y) \cup (x \cap (Obs_1 \setminus Obs_{12}))) \cup ((y \cap z) \cup (z \cap (Obs_2 \setminus Obs_{12}))) \mid x \in a_1.\lambda,\ y \in a_{12}.\lambda$ *and* $z \in a_2.\lambda\}$;
2. $a.E = a_1.E \cup a_2.E$

Note first that $a_{12}.\lambda$ is a singleton, that $Obs_i \cap Obs_{12}$, for $i = 1, 2$, is not empty (because $\mathcal{N}_1$ and $\mathcal{N}_2$ are interface compatible) but $Obs_i$ is not necessarily a subset of $Obs_{12}$, and that $Obs_1 \cap Obs_2 = \{term\}$. When we compose $a_1$ and $a_2$, if $a_1$ (resp. $a_2$) can progress in $\mathcal{G}_1$ (resp. $\mathcal{G}_2$) by using local observed transitions (i.e., transitions in $Obs_1 \setminus Obs_{12}$ (resp. $Obs_2 \setminus Obs_{12}$)), the product aggregate $a$ should be able to do the same. If this is not the case, then $a$ has to have the same behavior as $a_1$ (resp. $a_2$) and $a_{12}$ conjointly.

**Definition 14.** *Let $\mathcal{G}_i = \langle \mathcal{A}_i, Obs_i, \rightarrow_i, a_{0i}, \Omega_i \rangle, i = 1, 2$ be two SOGs corresponding to two oWF-nets $\mathcal{N}_1$ and $\mathcal{N}_2$. Let $\mathcal{G}_{12} = \langle \mathcal{A}_{12}, Obs_{12}, \rightarrow_{12}, a_{012}, \Omega_{12} \rangle$ be the SOG of the medium net $N_{12}$. The composition of $\mathcal{G}_1$ and $\mathcal{G}_2$, namely $\mathcal{G}_1 \oplus \mathcal{G}_2 = \langle \mathcal{A}, Act, \rightarrow, a_0, \Omega \rangle$ is defined as follows:*

1. $\mathcal{A} \subseteq \mathcal{A}_1 \times \mathcal{A}_{12} \times \mathcal{A}_2$;
2. $Act = Obs_1 \cup Obs_2$;
3. $\rightarrow$ *is the transition relation, defined by:*
   $\forall (a_1, a_{12}, a_2) \in \mathcal{A},\ \forall (a'_1, a'_{12}, a'_2) \in \mathcal{A},\ (a_1, a_{12}, a_2) \xrightarrow{o} (a'_1, a'_{12}, a'_2) \Leftrightarrow$
   $$\begin{cases} a_1 \xrightarrow{o}_1 a'_1 \wedge a_{12} \xrightarrow{o}_{12} a'_{12} \wedge a'_2 = a_2 & \text{if } o \in (Obs_1 \cap Obs_{12}) \\ a'_1 = a_1 \wedge a_{12} \xrightarrow{o}_{12} a'_{12} \wedge a_2 \xrightarrow{o}_2 a'_2 & \text{if } o \in (Obs_2 \cap Obs_{12}) \\ a_1 \xrightarrow{o}_1 a'_1 \wedge a'_{12} = a_{12} \wedge a'_2 = a_2 & \text{if } o \in (Obs_1 \setminus Obs_{12}) \\ a'_1 = a_1 \wedge a'_{12} = a_{12} \wedge a_2 \xrightarrow{o}_2 a'_2 & \text{if } o \in (Obs_2 \setminus Obs_{12}) \end{cases}$$
4. $a_0 = (a_{01}, a_{012}, a_{02})$;
5. $\Omega = \Omega_1 \times \Omega_{12} \times \Omega_2$.

The composition of the SOGs is similar to the classical synchronized product between graphs, except the fact that nodes are aggregates (carrying additional information) instead of single states. However, the asynchronous composition of the corresponding oWF-nets has been reduced to a synchronous composition

involving the medium net. The evolution in $\mathcal{G}_1 \oplus \mathcal{G}_2$ can stand for a local evolution to $\mathcal{G}_1$ (resp. $\mathcal{G}_2$) by using point 3 (resp. 4) of the transition relation in Definition 14, or a simultaneous evolution in $\mathcal{G}_1$ (resp. $\mathcal{G}_2$) and $\mathcal{G}_{12}$ by using point 1 (resp. 2). Given a local transition $t$ in $\mathcal{N}_1$, for instance, one can check whether it remains enabled after composition or not. Indeed, the union of the $E$ attribute of each aggregate $a_1$, being a part of an aggregate of $\mathcal{G}_1 \oplus \mathcal{G}_2$, should contain $t$. Otherwise, the transition $t$ is not enabled by the composite net and the transition $t$ becomes dead in the composition. If all the local transitions remain enabled, the other requirements of soundness can be deduced by analyzing the synchronized product of the SOGs.

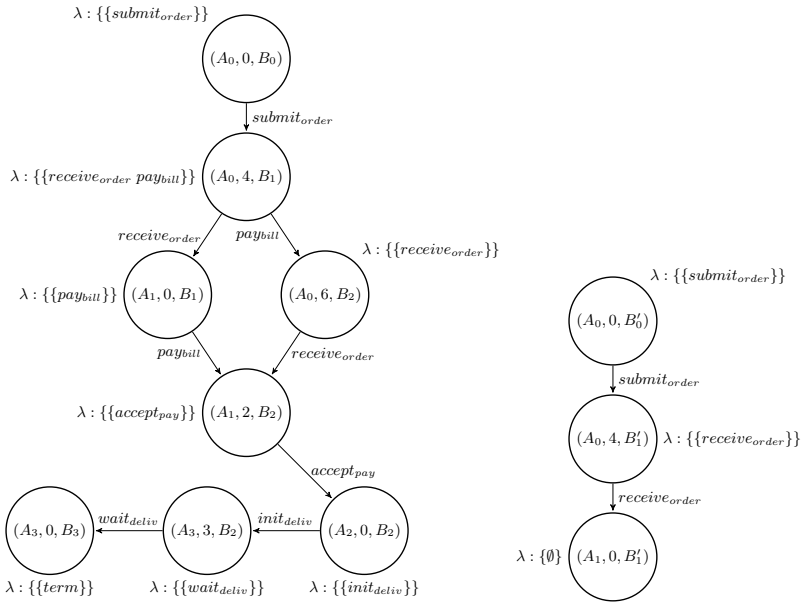



**Fig. 5.** the SOG synchronized product

Figure 5 illustrates the two SOGs obtained by synchronizing the SOG of online shop of Figure 2(a) with the SOGs of costumer $C_1$ and $C_2$ of Figure 2.

**Theorem 2.** *Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two oWF- nets and let $\mathcal{G}_1$ and $\mathcal{G}_2$ be the corresponding SOGs respectively. Then, $\mathcal{G}_1 \oplus \mathcal{G}_2$ is a SOG of $\mathcal{N}_1 \oplus \mathcal{N}_2$ with respect to $Obs_1 \cup Obs_2$.*

### 5.2 Checking Services Compatibility

Our goal is to check compatibility between two interface compatible oWF-nets $\mathcal{N}_1$ and $\mathcal{N}_2$ using their respective SOGs $\mathcal{G}_1$ and $\mathcal{G}_2$. We assume that the two oWF-nets are already sound. For checking compatibility, we have to check the

soundness property of $\mathcal{N}_1 \oplus \mathcal{N}_2$. This verification will be reduced to the analysis of the synchronized product of $\mathcal{G}_1$ and $\mathcal{G}_2$, denoted $\mathcal{G}_1 \oplus \mathcal{G}_2$.

**Theorem 3.** *Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two oWF-nets locally sound and let $\mathcal{G}_1$ and $\mathcal{G}_2$ be the corresponding SOGs respectively. Assume that all the local transitions remain enabled in the composition $\mathcal{G}_1 \oplus \mathcal{G}_2$. Then, $\mathcal{N}_1 \oplus \mathcal{N}_2$ is sound $\Leftrightarrow$*

1. *for each aggregate $a$ in $\mathcal{G}_1 \oplus \mathcal{G}_2$, $\emptyset \notin a.\lambda$,*
2. *for each aggregate $a$ in $\mathcal{G}_1 \oplus \mathcal{G}_2$, $\exists$ a final aggregate $a_f$ such that $a_f \in R(a)$,*
3. *for each observed transition $t$, $\exists$ two aggregates $a$, $a'$ in $\mathcal{G}_1 \oplus \mathcal{G}_2$ s.t. $a \xrightarrow{t} a'$.*

**Corollary 1.** *Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two oWF-nets and let $\mathcal{G}_1$ and $\mathcal{G}_2$ be the corresponding SOGs respectively. $\mathcal{N}_1$ is compatible with $\mathcal{N}_2 \Leftrightarrow \mathcal{G}_1 \oplus \mathcal{G}_2$ satisfies the three conditions of Theorem 3.*

By analyzing Figure 5, we can see that the composition of the online shop with the first customer is possible while it is not with the second: the corresponding composed SOG contains a deadlock (i.e., composite oWF-net not sound). For this particular example, checking the soundness property on the composition of SOGs (4 nodes and 3 edges) is easier than analyzing the original reachability graph which contain 24 nodes and 32 edges.

## 6 Related work

Several approaches investigated the issue of Web services composition. Even with emergence of web service process technologies such as industrial language BPEL4WS, WSCL, etc, this specification is still not the most suitable for the verification process of compatibility behavior on composition of web services. Thus, many researchers have been interested in formal modeling and analyzing methods to better formalize the behavior of web services such as Petri net model and its variants. Authors in [5] propose a Petri net-based Algebra for modeling web services control flows. The model is expressive enough to capture the semantics of complex service combinations. Formal semantics of each composition operator (e.g. sequence, selection, refinement) is expressed by a Petri net. Using this mechanism, the analysis of web services supports the verification of web services composition by checking properties like correct termination. An other technique for modeling multiple web services interactions between BPEL processes is discussed in [19] using an extention of Petri net models called composition net (C-net). Authors analyze the model through structural properties instead of the reachability states space in order to check compatibility: the compatibility is ensured when the composite net contains a non empty minimal siphon. They impose constraints on the model to prevent it from reaching incompatible cases by using a corresponding policy based on appending additional information to channels. Then, these channels are transformed back to a BPEL description so that a new compatible web service is obtained. An other approach [3] based on mediation aided composition has been widely adopted when dealing with incompatibilities of services. In this work, given two services modeled by oWF-net, the

authors propose to compose them using Mediation Transitions (MTs). They serve as information channel specifying the transferring relation of messages between different services. Then composition compatibility is verified by automatically constructing and analyzing the modular reachability graph (MRG) of the composition which is an abstraction of the original state graph. It is true that the performance of this approach is notable compared to classical ones, but MRG is represented explicitly which can be expensive.

Finally a similar approach has been introduced in [18]. In this work, the authors present a technique based on the Operating Guideline [14] for automatically checking accordance between a private view and a public view associated to each service involved in the overall process (composition of partners). A multiparty contract is specified in order to define the rules of engagement of each partner without describing its internal behavior. It can be seen as the composition of the public views from all partners. Based on the resulting contract, all participants implement their private view on the global process in such a way that it agrees with the contract. Then, checking accordance guarantees that the process is deadlock-free and that it will always terminate properly. The main differences with our approach are: (1) this approach works only for oWF-net with acyclic behaviors (and hence deadlock freedom coincides with weak termination), (2) It is an up-down approach in the sens that it starts from a public composition (contract) whose components can be modified locally under constraints. In our case, each component ignores all about the possible partners and we also allow local changes as long as the SOG is not modified. Finally, this approach uses operating guidelines [14] to abstract services and we established in [8] that, for most cases, the SOGs-based approach is more effective in terms of memory and time consumption. In conclusion, to the best of our knowledge, none of the existing approaches combine symbolic (using BDDs) abstraction and modular verification to check the compatibility of services. They always deal with an explicit representation of the system's behavior, which accentuate the state space explosion problem.

## 7 Conclusion

In this paper, we proposed an approach based on a suitable model, namely Symbolic Observation Graph, to abstract web services and to analyze their composition. Such an abstraction allow to respect the privacy of each publisher by hiding service's details, and at the same time it represents the necessary information to expose on a repository for possible collaboration with other web services. We established that and how symbolic observation graphs can be extended and efficiently used for that purpose. Using such abstraction, checking compatibility between two web services (a requester and a provider) is reduced to checking compatibility on the synchronized product of the corresponding SOGs.

We are currently developing a graph-based registry for abstract web services advertisement and discovery. the next step would be to extend the presented work in order (1) to deal with other compatibility criteria (e.g., other variants

of soundness, specific properties expressed with temporal logics, ...) and (2) to deal with richer models (e.g. shared resources, the explicit time).

## References


1. V. D. Aalst. The application of petri nets to workflow management, 1998.
2. R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.
3. Y. Du, X. Li, and P. Xiong. A petri net approach to mediation-aided composition of web services. *IEEE T. Automation Science and Engineering*, 9(2):429–435, 2012.
4. S. Haddad, J.-M. Ilié, and K. Klai. Design and evaluation of a symbolic and abstraction-based model checker. In *ATVA*, pages 196–210, 2004.
5. R. Hamadi and B. Benatallah. A petri net-based model for web service composition. In *ADC '03*, pages 191–200, 2003.
6. S. Hinz, K. Schmidt, and C. Stahl. Transforming bpel to petri nets. In *(BPM'05), volume 3649 of LNCS*, pages 220–235. Springer-Verlag, 2005.
7. K. Klai and J. Desel. Checking soundness of business processes compositionally using symbolic observation graphs. In *FMOODS/FORTE*, volume 7273 of *Lecture Notes in Computer Science*, pages 67–83. Springer, 2012.
8. K. Klai and H. Ochi. Modular verification of inter-enterprise business processes. In *eKNOW*, pages 155–161, 2012.
9. K. Klai and L. Petrucci. Modular construction of the symbolic observation graph. In *ACSD*, pages 88–97, 2008.
10. K. Klai and D. Poitrenaud. MC-SOG: An LTL model checker based on symbolic observation graphs. In *Petri Nets*, pages 288–306, 2008.
11. K. Klai, S. Tata, and J. Desel. Symbolic abstraction and deadlock-freeness verification of inter-enterprise processes. In *BPM'09 of LNCS*, pages 294–309. Springer-Verlag, 2009.
12. K. Klai, S. Tata, and J. Desel. Symbolic abstraction and deadlock-freeness verification of inter-enterprise processes. In *BPM'09 of LNCS*, pages 294–309, 2009.
13. N. Lohmann, P. Massuthe, C. Stahl, and D. Weinberg. Analyzing interacting bpel processes. In *BPM'06 of LNCS*, pages 17–32. Springer-Verlag, 2006.
14. P. Massuthe, W. Reisig, and K. Schmidt. An operating guideline approach to the soa. *Annals Of Mathematics, Computing and Teleinformatics*, 1:35–43, 2005.
15. C. A. Petri. Concepts of net theory. In *MFCS'73*, pages 137–146. Mathematical Institute of the Slovak Academy of Sciences, 1973.
16. C. Stahl, P. Massuthe, and J. Bretschneider. Transactions on petri nets and other models of concurrency ii. chapter Deciding Substitutability of Services with Operating Guidelines, pages 172–191. Springer-Verlag, 2009.
17. W. van der Aalst, K. van Hee, A. ter Hofstede, N. Sidorova, H. Verbeek, M. Voorhoeve, and M. Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing: applicable formal methods*, 23(3):333–363, 2010.
18. W. M. P. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl, and K. Wolf. Multiparty contracts: Agreeing and implementing interorganizational processes. *Comput. J.*, 53(1):90–106, 2010.
19. P. Xiong, Y. Fan, and M. Zhou. A petri net approach to analysis and composition of web services. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, pages 376–387, 2010.