

# A Behavioural Congruence for Web services<sup>\*</sup>

Filippo Bonchi, Antonio Brogi, Sara Corfini, Fabio Gadducci

Department of Computer Science – University of Pisa, Italy

**Abstract.** Web services are emerging as a promising technology for the development of next generation distributed heterogeneous software systems. We define a new behavioural equivalence for Web services, based on bisimilarity and inspired by recent advances in the theory of reactive systems. The proposed equivalence is compositional and decidable, and it provides a firm ground for enhanced behaviour-aware discovery and for a sound incremental development of services and service compositions.

## 1 Introduction

Web services are emerging as a promising technology for the development of next generation distributed heterogeneous software systems [16]. Roughly, a Web service is any piece of software that makes itself available over the Internet. A Web service is identified by a URI, it is universally accessible by means of standard protocols (WSDL, UDDI, SOAP), and it self-describes its functionalities by exposing a public interface.

WSDL [25] is the currently employed standard for describing services. A WSDL description details what a service provides, by listing its operations in terms of input and output messages, and how a service can be invoked, by specifying one or more network locations where it can be accessed. WSDL descriptions do not include any information on the interaction *behaviour* of services, that is, on the order with which messages can be received or sent by each service. Unfortunately, the lack of behavioural information inhibits the possibility of *a priori* determining whether two services have the same behaviour as well as the possibility of verifying properties of service compositions, such as deadlock-freedom.

During the last years, various proposals have been put forward to feature more expressive service descriptions that include both semantics (viz., ontology-based) and behaviour information about services. One of the major efforts in this direction is OWL-S [6], a high-level ontology-based language for describing services, proposed by the OWL-S coalition. Since OWL-S is a computer-interpretable semantic mark-up language providing all the needed information for describing services, OWL-S paves the way for the full automation of service discovery, invocation and composition. In particular, OWL-S service descriptions include a declaration of the interaction behaviour of services (the so-called *process model*), which provides the needed information for the *a priori* analysis and verification of service invocations and compositions.

---

<sup>\*</sup> Research partially supported by the EU FP6-IST IP 16004 SENSORIA and STREP 033563 SMEPP, and the MIUR FIRB TOCAI.It and PRIN 2005015824 ART.

The objective of this paper is to define a notion of *behavioural equivalence* for Web services. An immediate application of such a notion is the possibility of establishing whether syntactically different services feature the same behaviour, and hence for instance of verifying whether the upgrade of a service  $S$  with a new version  $S'$  may affect the interoperability with existing clients of  $S$ . The availability of a well-founded notion of behavioural equivalence can also be exploited to develop enhanced service discovery techniques so as to go beyond functional matching and determine whether a given service (or service composition) features a desired interaction behaviour. According to these aims, two fundamental properties of any equivalence relation are *computability* and *compositionality*. Computability is a key requirement in ensuring the viability of an equivalence relation, that is, in allowing the development of automated software capable of determining whether two services are behaviourally equivalent or not. Compositionality permits to exploit the equivalence relation for a disciplined incremental development of services, by means of sound compositions and replacements.

In this paper we first show how the behaviour of a Web service can be suitably described by means of a *Petri net*. Petri nets [19] are one of the best known and most widely adopted formalisms to express the concurrent behaviour of (software) systems: besides providing a clear and precise semantics, they feature an intuitive graphical notation, and a number of techniques and tools for their analysis, simulation and execution are available. Petri nets have also been already employed to model Web services (e.g., see [2, 11, 22]). We introduce a simple variant of standard condition/event Petri nets (viz., CPR nets for Consume-Produce-Read nets) to naturally model the behaviour of Web services, and in particular the persistence of data. We then show how OWL-S process models can be directly mapped into CPR nets, borrowing from the translation from OWL-S to place-transition nets (P/T for short) described in [5].

Our next step is the identification of a suitable behavioural equivalence for our class of nets. Indeed, the dynamics of a Petri net, as well as those of most functional and process calculi, is usually defined in terms of reduction relations among its markings (i.e., the states of a system). Despite its simplicity, the main drawback of reduction semantics is poor compositionality, in the sense that the dynamic behaviour of an arbitrary stand alone net may become unpredictable whenever it becomes a part of a larger net. Recently, Leifer and Milner [10] devised a methodology for distilling from a reduction relation a set of labels satisfying suitable requirements of minimality, in order to build a *labelled* reduction relation, such that the associated behavioural notion of *bisimilarity* is a compositional equivalence. The methodology has been later applied to P/T nets [14] as well as to their condition/event variant [21] (C/E for short). Thus, after discussing a motivating example, we define a novel notion of net equivalence, based on bisimilarity and inspired by the recent theoretical advances we just mentioned above. We show that this new equivalence relation is indeed compositional (i.e., that is a *congruence*), and that it is also decidable.

The main contribution of this paper is therefore the definition of a decidable behavioural congruence for Web services, which paves the way for the deployment of behaviour-aware discovery mechanisms and for a sound incremental development of services and service compositions.

## 2 Modeling Web Services with Petri nets

Before describing how Petri nets can be employed to model Web services, we recall the essence of OWL-S, a high-level ontology-based language for describing Web services. An OWL-S service advertisement consists of three documents: the *service profile*, providing a high-level specification of the service by describing its functional (i.e., inputs and outputs) and extra-functional attributes; the *process model*, describing the service behaviour by providing a view of the service in terms of process composition; and the *service grounding*, stating how to interact with the service by specifying protocol and message format information.

In the paper we focus on the OWL-S process model, as it details the behavioural information needed to represent a service as a Petri net. More precisely, the process model describes a service as a composite process which consists, in turn, of composite processes and/or atomic processes. An atomic process can not be decomposed further and it has associated inputs and outputs, while a composite process is built up by using a few control constructs: **sequence** (i.e., sequential execution), **choice** (conditional execution), **split** (parallel execution), **split+join** (parallel execution with synchronization), **any-order** (unordered sequential execution), **repeat-while** and **repeat-until** (iterative execution).

In [5] the second and third author proposed a mapping from OWL-S service descriptions to P/T nets with the objective to exploit the Petri net representation of services for a behaviour-aware service discovery. Given a query specifying the inputs and outputs of the service to be found, first the functional attributes of the available services are considered, in order to discover a composition capable of satisfying the query functionally. Next, the found composition is translated into a P/T net and analysed in order to verify properties such as deadlock-freedom.

The mapping presented in [5] takes into account the fact that an OWL-S atomic operation can be executed only if all its inputs are available and all the operations that must occur before its execution have been completed. Hence, atomic operations are mapped into transitions, and places and transition firing rules are employed to model both the availability of data (i.e., the *data flow*) and the executability of atomic operations (i.e., the *control flow*). Indeed, an atomic operation  $T$  is modelled as a transition  $t$  having an input/output *data place* for each input/output of  $T$ , an *input control place* to denote that  $t$  is executable as well as an *output control place* to denote that  $t$  has completed its execution.

However, the specific features concerning the Web service framework (already emphasised in [5]) suggest more specific nets. In this paper we introduce a simple variant of standard Petri nets [19] as a tool for properly modelling Web services. Given a net  $N$  modelling a Web service, we first of all noted that the portion of  $N$  restricted to transitions and control places should behave as a classical C/E net, that is, only one token should occur at most for each place. Furthermore, we opted for a model stressing the persistence of data, meaning that, once a data has been produced by some service operation, it has to be kept available for all the service operations that input it. In other words, whilst control places can be produced and consumed, data places can be read, produced but not consumed. Hence, the portion of  $N$  restricted to data places behaves as a *contextual* net [15].

To encompass into the same structure the different net behaviour determined by data and control places, we introduce our own particular flavour of contextual C/E nets, which is going to be formally introduced in the following subsection.

## 2.1 Consume-Produce-Read nets

This subsection introduces *consume-produce-read* nets. These are a slight extension of standard Petri nets, since they are equipped with two disjoint sets of places, namely, the *control* (for consume-produce) places and the *data* (for produce-read) places.

**Definition 1 (CPR net).** A consume-produce-read net (*simply*, CPR net)  $N$  is a five-tuple  $(C_N, D_N, T_N, F_N, I_N)$  where

- $C_N$  is a finite set of control places,
- $D_N$  is a finite set of data places (*disjoint from*  $C_N$ ),
- $T_N$  is a finite set of transitions,
- $F_N \subseteq (C_N \times T_N) \cup (T_N \times C_N)$  is the control flow relation,
- $I_N \subseteq (D_N \times T_N) \cup (T_N \times D_N)$  is the data flow relation.

Our nets behave as standard C/E nets with respect to control places; while, as we are going to see, data places are never emptied, once they are inhabited.

As for standard nets, we associate a *pre-set* and a *post-set* with each transition  $t$ , together with two additional sets, called *read-* and *produce-set*.

**Definition 2 (pre-, post-, read-, and produce-set).** Given a CPR net  $N$ , we define for each  $t \in T_N$  the sets

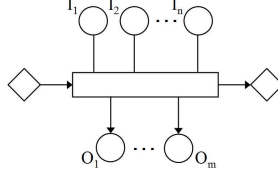
$$\begin{aligned} \diamond t &= \{s \in C_N \mid (s, t) \in F_N\} & t^\diamond &= \{s \in C_N \mid (t, s) \in F_N\} \\ \bullet t &= \{s \in D_N \mid (s, t) \in I_N\} & t^\bullet &= \{s \in D_N \mid (t, s) \in I_N\} \end{aligned}$$

which denote respectively the pre-set, post-set, read-set and produce-set of  $t$ .

Fig. 1 depicts our chosen graphical notation. Diamonds represent control places, while circles and rectangles represent data places and transitions, respectively. For instance, the transition shown in Fig. 1 *reads* the data places labelled  $I_1, I_2, \dots, I_n$  (this is represented by a straight line) and *produces* the data places labelled  $O_1, \dots, O_m$  (this is represented by a pointed arrow). In doing so, the control flow passes from the left-most to the right-most control place.

**Definition 3 (marking).** Given a CPR net  $N$ , a marking  $M$  for  $N$  is a finite set of places in  $P_N = C_N \cup D_N$ .

A marking of the net  $N$  coincides with a subset of its set  $P_N$  of places, since each place can contain at most one token. The evolution of a net is given by a relation over markings. A transition  $t$  is *enabled* by a marking  $M$  if the control places which belong to the pre-set of  $t$  as well as the data places which belong to the read-set of  $t$  are contained in  $M$ , and no overlap (as defined later) between  $M$  and the post-set of  $t$  occurs. In this case a *firing step* may take place:  $t$  removes the tokens from the control places which belong to the pre-set of  $t$  and adds a token to each place which belongs to the post- and produce-set of  $t$ .



**Fig. 1.** Modelling atomic operations as CPR net transitions.

**Definition 4 (firing step).** Let  $N$  be a CPR net. Given a transition  $t \in T_N$  and a marking  $M$  for  $N$ , a firing step is a triple  $M[t]M'$  such that  $(\circ t \cup \bullet t) \subseteq M$  and  $(M \cap t^\circ) \subseteq \circ t$  ( $M$  enables  $t$ ), and moreover  $M' = (M \setminus \circ t) \cup t^\circ \cup \bullet t$ .

We write  $M \setminus M'$  if there exists some  $t$  such that  $M[t]M'$ .

The enabling condition states that (i) all the tokens of the pre-set of a transition have to be contained in the marking, and (ii) that the marking does not contain any token in the post-set of the transition, unless it is consumed and regenerated. The second condition usually characterizes C/E nets. Note instead that data places act as sinks, hence, any token can be added and only the occurrence of a token is checked. The read-only feature of data places is reminiscent of the work on so-called *contextual* C/E nets by Montanari and Rossi [15].

## 2.2 From OWL-S to CPR nets

After formally defining CPR nets, we can show how OWL-S service descriptions can be mapped into CPR nets.

Let us define a service as a triple  $(i, P, f)$  where  $P$  denotes the CPR (sub)net representing the service and  $i$  and  $f$  denote the initial and the final control places of  $P$ , respectively. To define compositional operators is sufficient to properly coordinate the initial and final control places of the employed services. For instance, let us consider the *sequential composition* of two services  $(i_1, P1, f_1)$  and  $(i_2, P2, f_2)$ . This is a CPR net consisting of the two services and a transition whose starting control place is  $f_1$  and the final control place is  $i_2$ . By doing so,  $P1$  has to be completed before  $P2$  can start.

For lack of space, we do not give a formal semantics of the sequence operator and of the other OWL-S control constructs. Anyway, the reader can intuitively understand how OWL-S composite operations can be mapped into CPR nets by observing Fig. 2. The  $PX$ -labelled boxes represent  $(i_x, PX, f_x)$  services, the dark gray rectangles identify empty transitions, and the light gray diamonds denote the starting and final control places of the resulting nets. To simplify reading we omitted data places from the nets of Fig. 2. Yet, it is important to note that the  $PX$ -boxes can share data places to simulate the exchange of data amongst services, as we will formally define in Section 4.1.

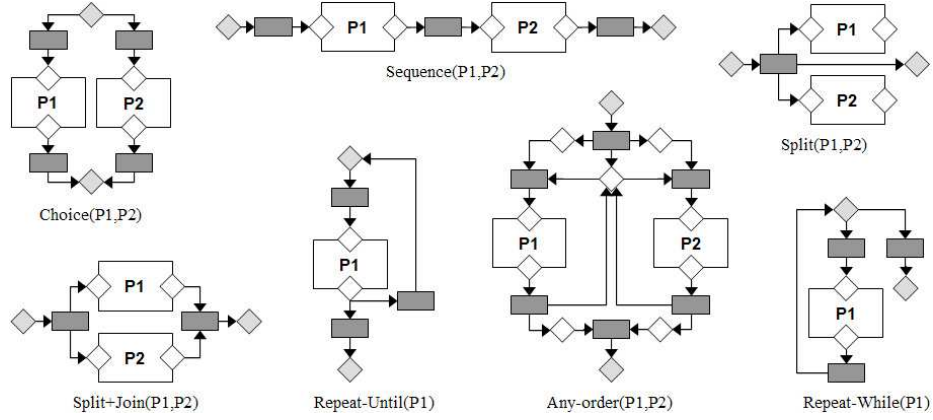


Fig. 2. Modelling OWL-S composite operations as CPR nets.

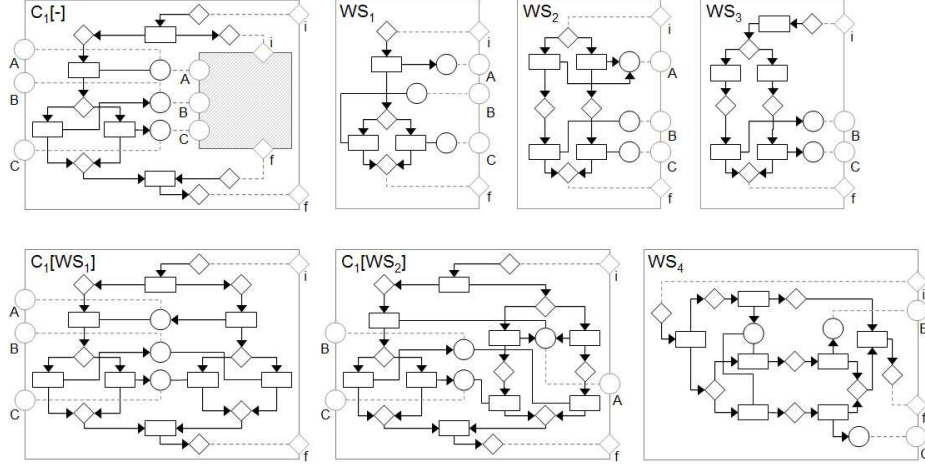
### 3 Motivating Examples

The ultimate objective of this paper is to introduce a decidable notion of equivalence between Web services represented as CPR nets. This notion establishes whether two service behaviours are equivalent and it can be employed to suitably address the following issues:

- 1) *incremental development of services* — to check whether two different versions of a service are equivalent;
- 2) *matching services* — to check whether a (composition of) service(s) matches a query that specifies the behaviour of the desired service (composition) to be found (viz., the notion of equivalence needed by [5]);
- 3) *replaceability of services* — to check whether a service  $s$  which takes part in a composition  $C[s]$  can be replaced with a different service  $r$  without changing the behaviour of  $C$ , i.e., guaranteeing the equivalence between  $C[s]$  and  $C[r]$ .

We can hence outline the main features that a suitable notion of equivalence should have, that is, *weakness* and *compositionality*. It has to be weak as it must equate services with respect to their externally observable behaviour. Indeed, it is reasonable that two versions of a service differently implement the same operations (1), as well as we can imagine that a simple query can be satisfied by a complex service composition (2). Therefore, this notion of equivalence has to be abstract enough to equate services that differ only on internal transition steps. Furthermore, (3) also asks for compositionality, and if two services are equivalent, then they can be always used interchangeably.

Let us now consider some examples (inspired by [12]). Fig. 3 illustrates the CPR nets of seven services, where rectangles, circles and diamonds represent transitions, data places and control places, respectively. Note the boxes that limit each service. As we will formalise in the following section, a box represents the *outer* interface of a service, that is, the set of places which can interact



**Fig. 3.** Example of (non-)equivalent services.

with the environment. Hence, those places that in Fig. 3 lie on the box are the ones that can be observed externally. Consider the services  $WS_1$  and  $WS_2$  of Fig. 3. As one may note,  $WS_1$  and  $WS_2$  have the same behaviour with respect to the notion of *trace equivalence*. Indeed, they have identical sets of traces, since after producing  $A$  they may alternatively read either  $B$  or  $C$ . Consider now the context  $C_1[-]$ , depicted in Fig. 3, which represents a possible environment in which  $WS_1$  and  $WS_2$  can be embedded. Note the gray area contained in  $C_1[-]$ . Its border is the *inner* interface of  $C_1[-]$ . As formalised in Section 4.1, a service  $WS$  can be inserted inside a context  $C[-]$  if the outer interface of  $WS$  and the inner interface of  $C[-]$  coincide. The resulting composition  $C[WS]$  consists of the fusion of such interfaces, as well as of the fusion of the data places and the union of the transitions of  $WS$  and  $C[-]$ . We note that  $C_1[-]$  inputs  $A$ , produced by  $WS_1$ , and yields  $B$  or  $C$ , taken as input by  $WS_1$ . Hence, the composition  $C_1[WS_1]$  works and finishes properly. Now, in order to test if the trace equivalence is the notion suitable for our purpose, we replace  $WS_1$  with the trace equivalent service  $WS_2$  and we check whether the composition  $C_1[WS_2]$  works properly as well. Yet,  $C_1[WS_2]$  produces a possibly deadlocking system.

Let us now describe a second example arguing the need of weakness. Consider the services  $WS_3$  and  $WS_4$ . For instance,  $WS_4$  could be a composition candidate to satisfy the query represented by  $WS_3$ . Although  $WS_3$  and  $WS_4$  appear different as they perform a different number of transitions, they both produce  $B$  or  $C$ . Namely,  $WS_3$  and  $WS_4$  have identical externally observable behaviour, and they indeed should be considered equivalent.

By taking into account the requirements briefly outlined in this motivating section, we define a novel notion of equivalence based on bisimilarity which features weakness, compositionality and decidability.

## 4 A congruence for Open CPR nets

In the previous section we argued about the relevance of a behavioural equivalence, formally characterizing the notion of replaceability, and we motivated why such an equivalence should be both compositional and weak.

A first step for defining compositionality is to equip nets with a notion of interface and context. Next, we introduce two notions of equivalence: the first is conceptually the correct one, even if it turns out to be quite hard to reason about, while the second provides a simple, decidable characterization of the former.

### 4.1 Open CPR nets and CPR contexts

For the sake of presentation, a chosen net  $N = (C_N, D_N, T_N, F_N, I_N)$  is assumed.

**Definition 5 (Open CPR net).** *Let  $N$  be a CPR net. An interface for  $N$  is a triple  $\langle i, f, OD \rangle$  such that*

- $i$  is a control place (i.e.,  $i \in C_N$ ), the initial place;
- $f$  is a control place (i.e.,  $f \in C_N$ ), the final place;
- $OD$  is a set of data places (i.e.,  $OD \subseteq D_N$ ), the open data places.

An interface is an outer interface  $O$  for  $N$  if there exists no transition  $t \in T_N$  such that either  $i \in t^\circ$  or  $f \in \circ t$ . An open CPR net  $\mathcal{N}$  (OCPR for short) is a pair  $\langle N, O \rangle$ , for  $N$  a CPR net and  $O$  an outer interface for  $N$ .

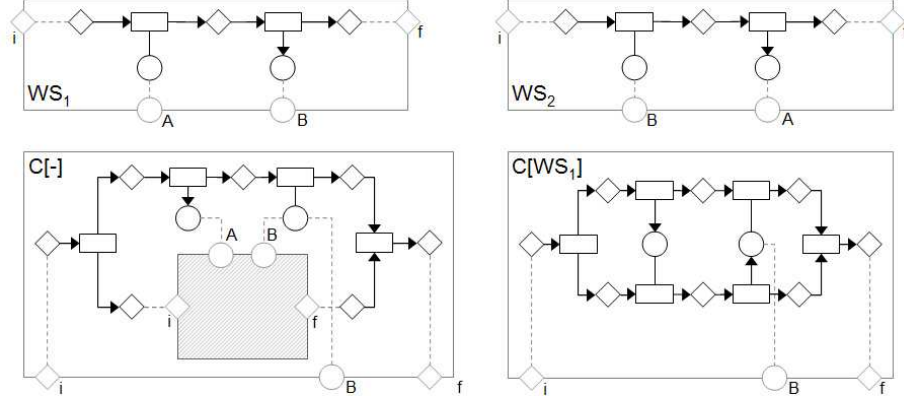
The condition characterizing outer interfaces requires that the initial place has no incoming transition and the final place has no outgoing transition. Hence, OCPR nets recall the Work-Flow (WF) nets proposed by van der Aalst [24]<sup>1</sup>.

Given an open net  $\mathcal{N}$ ,  $Op(\mathcal{N})$  denotes the set of *open places*, which consists of those places occurring on the interface, initial and final places included. Furthermore, the places of  $\mathcal{N}$  not belonging to  $Op(\mathcal{N})$  constitute the *closed places*. It is important to note that open places are crucial for the definition of observational equivalence that we propose in the next subsection.

Fig. 4 shows the graphical notation used to represent OCPR nets. The bounding box of the OCPR net  $WS_1$  represents the outer interface of the net. Note the initial and final places used to compose the control of services (as suggested in Fig. 2) as well as the open data places employed to share data. Next, we symmetrically define an *inner interface* for  $N$  as an interface such that there is no transition  $t \in T_N$  verifying either  $f \in t^\circ$  or  $i \in \circ t$ .

**Definition 6 (CPR contexts).** *Let  $N$  be a CPR net. A CPR context  $C[-]$  is a triple  $\langle N, O, I \rangle$ , where  $N$  is a CPR and  $I$  and  $O$  are an inner and an outer interface for  $N$ , respectively.*

<sup>1</sup> More precisely, our nets lack the connectiveness requirement, see e.g. [24, Sect. 2.2]. Note however that, even if this is not made explicit, the property holds for all the nets obtained by modeling OWL-S composite operations.



**Fig. 4.** Two open nets, a context and a composite net.

A context  $C[-]$  is shown in Fig. 4. Substantially, it is an open net with a hole, represented there by a gray area. The border of the hole denotes the inner interface of the context. As for open nets, the bounding box is the outer interface. The only difference between inner and outer interfaces is that the initial place of the former has no outgoing transitions (and vice versa for final places).

Contexts represent environments in which services can be embedded, i.e., possible ways they can be used by other services. An OCPN net can be inserted in a context if the net outer interface and the context inner interface coincide.

**Definition 7 (CPR composition).** *Let  $\mathcal{N} = \langle N, O \rangle$  be an OCPN net and  $C[-] = \langle N_C, O_C, I_C \rangle$  a CPR context, such that  $O = I_C$ . Then, the composite net  $C[\mathcal{N}]$  is the OCPN net  $(C_N \uplus_O C_{N_C}, D_N \uplus_O D_{N_C}, T_N \uplus T_{N_C}, F_N \uplus F_{N_C}, I_N \uplus I_{N_C})$  with outer interface  $O_C$ .*

In other words, the disjoint union of the two nets is performed, except for those places occurring in  $O$ , which are collapsed: this is denoted by the symbol  $\uplus_O$ . Moreover,  $O_C$  becomes the set of open places of the resulting net.

Consider e.g. the net  $WS_1$ , the context  $C[-]$  and their composition, denoted by  $C[WS_1]$ , as illustrated in Fig. 4. The places on the outer interface of  $WS_1$  are coalesced with the ones on the inner interface of  $C[-]$ . The output interface of  $C[WS_1]$  is the outer interface of  $C[-]$ . Note that the data place  $A$  is open in  $WS_1$  and closed in  $C[WS_1]$ : this example highlights the capability of hiding places, removing them from the outer interface of an open net. Indeed, this feature is reminiscent of the restriction operator of process calculi, such as CCS [13].

It is worth observing that contexts can be composed with contexts as well: CPR contexts form a category where interfaces are objects and contexts are arrows going from the inner interface to the outer interface (and OCPN nets are arrows whose source is the empty interface and target is the outer interface)<sup>2</sup>.

<sup>2</sup> CPR nets and their morphisms (not defined here) form an *adhesive category* [9]. The category of CPR contexts is the *cospan bicategory* over such category [20], thus it is

## 4.2 Saturated bisimilarity for OCPR nets

This section addresses the question of the equivalence between nets. Our answer relies on an observational approach, equating two systems if they can not be told apart from an external observer. More precisely, the observer can only examine the open places of a net, which is otherwise a black box, and only for those places he may check if they are actually inhabited or if they are empty.

For the sake of presentation, a chosen OCPR net  $\mathcal{N} = \langle N, O \rangle$  is assumed.

**Definition 8 (observation).** *Let  $\mathcal{N}$  be an OCPR net, and  $M$  a marking of  $N$ . The observation on  $\mathcal{N}$  at  $M$  is the set of places  $Obs(\mathcal{N}, M) = Op(\mathcal{N}) \cap M$ .*

Thus, an observer can look the evolution of the system by observing if tokens are produced or consumed in the open places. Accordingly, two OCPR nets  $\mathcal{N}$  and  $\mathcal{N}'$  with initial markings  $M$  and  $M'$  are going to be considered equivalent if  $Obs(\mathcal{N}, M) = Obs(\mathcal{N}', M')$  and if every state reachable from  $M$  in  $N$  is equivalent to a state reachable from  $M'$  in  $N'$  (and vice versa).

The previous remark is formalized by the following definition, where  $\mathcal{MN}$  denotes the set of all OCPR nets with markings and  $\rightarrow_{\mathcal{N}}$  denotes the reflexive and transitive closure of the firing relation  $\lrcorner$  for the net  $N$  underlying  $\mathcal{N}$ .

**Definition 9 (naive bisimulation).** *A symmetric relation  $\mathfrak{R} \subseteq \mathcal{MN} \times \mathcal{MN}$  is a naive bisimulation if whenever  $(\mathcal{N}, M) \mathfrak{R} (\mathcal{N}', M')$  then*

- $Obs(\mathcal{N}, M) = Obs(\mathcal{N}', M')$ , and
- $M \rightarrow_{\mathcal{N}} M_1$  implies  $M' \rightarrow_{\mathcal{N}'} M'_1$  and  $(\mathcal{N}, M_1) \mathfrak{R} (\mathcal{N}', M'_1)$ .

*The union of all naive bisimulations is called naive bisimilarity.*

The equivalence above is “naive” in the sense that it clearly fails to be compositional. Indeed, consider the OCPR nets  $WS_1$  and  $WS_2$  in Fig. 4. They are trivially equivalent since none of them can fire. However, they are not equivalent anymore whenever they are inserted into a context with a transition generating a token in the initial place and in the data place  $A$ . Indeed the former can now produce a token on  $B$  reaching the final state  $f$ , while the latter can not move.

The solution out of the impasse, which is quite standard both in functional languages and process calculi, is to allow the observer to perform more complex experiments, inserting a net into any possible context.

**Definition 10 (saturated bisimulation).** *A symmetric relation  $\mathfrak{R} \subseteq \mathcal{MN} \times \mathcal{MN}$  is a saturated bisimulation if whenever  $(\mathcal{N}, M) \mathfrak{R} (\mathcal{N}', M')$  then  $\forall C[-]$*

- $Obs(\mathcal{N}, M) = Obs(\mathcal{N}', M')$ , and
- $M \rightarrow_{C[\mathcal{N}]} M_1$  implies  $M' \rightarrow_{C[\mathcal{N}']} M'_1$  and  $(C[\mathcal{N}], M_1) \mathfrak{R} (C[\mathcal{N}'], M'_1)$ .

*The union of all saturated bisimulations is called saturated bisimilarity ( $\approx_S$ ).*

---

amenable to the borrowed context technique [7] for distilling a set of labels from a reduction relation. That technique is implicitly exploited in the next section.

Clearly,  $\approx_S$  is by definition a congruence. Indeed, it is the largest bisimulation that preserves compositionality, as stated below.

**Proposition 1.**  *$\approx_S$  is the largest bisimulation that is also a congruence.*

The above proposition ensures the compositionality of the equivalence, hence, the possibility of replacing one service by an equivalent one without changing the behaviour of the whole composite service. Moreover, the equivalence is “weak” in the sense that, differently from most of the current proposals, no explicit occurrence of a transition is observed. The previous definition leads to the following notion of equivalence between OCPN nets, hence, between services.

**Definition 11 (bisimilar nets).** *Let  $\mathcal{N}, \mathcal{N}'$  be OCPN nets. They are bisimilar, denoted by  $\mathcal{N} \approx \mathcal{N}'$ , if  $(\mathcal{N}, \emptyset) \approx_S (\mathcal{N}', \emptyset)$ .*

The choice of the empty marking guarantees that the equivalence is as general as possible. Indeed, the presence of a token in an open place can be simulated by closing the net with respect to a transition adding a token in that place, and if any two nets are saturated bisimilar with respect to the empty marking, they are so also with respect to any marking with tokens in the open places.

The negative side of  $\approx$  is that this equivalence seems quite hard to be automatically decided because of the quantification over all possible contexts. In the following subsection we introduce an alternative equivalence, easier to reason about and to automatically verify, and we prove that it coincides with  $\approx_S$ .

### 4.3 An equivalent decidable bisimilarity

Saturated bisimulation seems conceptually the right notion, and this is further argued in the following section. However, it seems quite hard to analyze (or automatically verify), due to the universal quantification over contexts. In this section we thus introduce *semi-saturated* bisimilarity, based on a simple *labelled transition system* (LTS) distilled from the firing semantics of an OCPN net.

The introduction of an LTS is inspired to the theory of *reactive systems* [10]. This meta-theory suggests guidelines for deriving a LTS from an unlabelled one, choosing a set of labels with suitable requirements of minimality. In the setting of OCPN nets, the reduction relation is given by  $[\ ]$ , and a firing is allowed if all the preconditions of a transition are satisfied. Thus, intuitively, the minimal context that allows a firing just adds the tokens needed for that firing.

**Definition 12 (labelled transition system).** *Let  $\mathcal{N}$  be an OCPN net, and let  $\Lambda = \{\tau\} \cup (\{+\} \times P_{\mathcal{N}}) \cup (\{-\} \times C_{\mathcal{N}})$  be a set of labels, ranged over by  $l$ . The transition relation for  $\mathcal{N}$  is the relation  $R_{\mathcal{N}}$  inductively generated by the set of inference rules below*

$$\frac{o \in Op(\mathcal{N}) \setminus (M \cup \{f\})}{M \xrightarrow{\tau}_{\mathcal{N}} M \cup \{o\}} \quad \frac{f \in M}{M \xrightarrow{f}_{\mathcal{N}} M \setminus \{f\}} \quad \frac{M [\ ] M'}{M \xrightarrow{\tau}_{\mathcal{N}} M}$$

where  $M \xrightarrow{l}_{\mathcal{N}} M'$  means that  $\langle M, l, M' \rangle \in R_{\mathcal{N}}$ , and  $i, f$  denote the initial and the final place of  $\mathcal{N}$ , respectively.

Thus, a context may add tokens in open places in order to perform a firing, as represented by the transition  $\xrightarrow{+o}_{\mathcal{N}}$ . Similarly, a context may consume tokens from the final place  $f$ . A context can not interact with the net in other ways, as the initial place  $i$  can be used by the context only as a post condition, as well as all the other open places are data places whose tokens can be read but not consumed. Instead,  $\tau$  transitions represent internal firing steps, i.e., steps that do not need any additional token from the environment.

The theory of reactive systems ensures that, for a suitable choice of labels, the bisimilarity on the derived LTS is a congruence [10]. However, often that bisimilarity does not coincide with the saturated one. In [4] the first author, together with König and Montanari, discusses this problem and introduces the notion of semi-saturated bisimilarity that coincides with the saturated one.

**Definition 13 (semi-saturated bisimulation).** *A symmetric relation  $\mathfrak{R} \subseteq \mathcal{MN} \times \mathcal{MN}$  is a semi saturated bisimulation if whenever  $(\mathcal{N}, M) \mathfrak{R} (\mathcal{N}', M')$*

- $Obs(\mathcal{N}, M) = Obs(\mathcal{N}', M')$ ,
- $M \xrightarrow{+o}_{\mathcal{N}} M_1$  implies  $M' \cup \{o\} \rightarrow_{\mathcal{N}'} M'_1$  and  $(\mathcal{N}, M_1) \mathfrak{R} (\mathcal{N}', M'_1)$ ,
- $M \xrightarrow{-f}_{\mathcal{N}} M_1$  implies  $M' \setminus \{f\} \rightarrow_{\mathcal{N}'} M'_1$  and  $(\mathcal{N}, M_1) \mathfrak{R} (\mathcal{N}', M'_1)$ ,
- $M \xrightarrow{\tau}_{\mathcal{N}} M_1$  implies  $M' \rightarrow_{\mathcal{N}'} M'_1$  and  $(\mathcal{N}, M_1) \mathfrak{R} (\mathcal{N}', M'_1)$ .

The union of all semi-saturated bisimulations is called semi-saturated bisimilarity ( $\approx_{SS}$ ).

The key theorem of the paper is stated below.

**Theorem 1.**  $\approx_S = \approx_{SS}$ .

Thus, in order to prove that two OCPR nets are bisimilar, it suffices to exhibit a semi-saturated bisimulation between the states of the two nets that includes the pair of empty markings. Most importantly, though, this search can be automatically performed, since the set of possible states of an OCPR net are finite. Hence, the result below immediately follows.

**Corollary 1.**  $\approx_S$  is decidable.

## 5 Related Works

The successful introduction of observational equivalences for process calculi in the early 1980s spawned similar researches on nets, as witnessed by the survey [18]<sup>3</sup>. According to the taxonomy there, saturated bisimilarity is a state-based equivalence, since it encompasses a notion of interface (a set of observable places) and it is dictated by the way the firing relation crosses the interfaces. Indeed, our bisimilarity is reminiscent of *ST-equivalence*, as in [18, Def. 4.2.6].

<sup>3</sup> The analysis there is restricted to *contact-free* C/E nets, i.e., such that the second requirement of the enabling condition in Definition 4 is never verified.

This section does not try to survey the field: first of all, the literature is very large, and its retelling is not suitable for a conference paper. Moreover, our CPR nets have distinctive features, since data places act as sinks, thus any comparison should take that fact into account, putting an additional layer of complexity. Furthermore, our main interest is in a *compositional* equivalence, thus restricting the area of possible intersection with former works.

In the rest of the section we then focus on two issues that are closely related to the novelties introduced in our framework: the use of the theory of reactive systems for obtaining a tractable equivalence; and the use of (equivalences on) nets for dealing with the specification of Web services and of their composition.

## 5.1 Nets, open places and labels

Most current-day formalisms for system specification come equipped with a reduction semantics: a suitable algebra of states is defined, and system evolution is represented by a relation between states. However, the lack of observable actions (either associated to the states, or to the reductions) forbids the development of observational equivalences, which are often handier and more tractable.

Concerning Petri nets, the need of primitives for expressing the interaction with an environment was recognized early on, and notions of “net interface”, intended as a subset of the items of the net, are already reported in [18]. Interfaces are key ingredients for defining an observation, as well as for expressing net operators: along this line, a classical approach is the Box Algebra [3]. The main difference with our proposal is in the use of a set of labels for obtaining a labelled reduction relation, on top of which to define the semi-saturated semantics.

Indeed, mostly related to our solution are *open nets*: place/transitions nets where two distinguished sets of input and output places (where tokens may be added or removed, respectively) are identified, and then used to compose nets by place coalescing [1]. A comparison among the derived equivalences is in order, even if it is left for future work: note however that the authors stick to P/T nets; and the dichotomy between input and output places, that is reflected on net composition, is missing in our approach. We refer to [1], and the references therein, for a survey and comparison with other interface-based techniques, mostly important the *net components* proposed by Kindler [8].

The most important source of inspiration for our work is the theory of reactive systems [10], introducing a technique for synthesising labels with suitable minimality requirements from a reduction relation that guarantees that the bisimilarity on the derived labelled relation is a congruence. Indeed, our approach benefits from the general definition of interface deriving from the theory. Concerning Petri nets, the technique was first applied by Milner in [14], after implementing nets into a more complex graphical structure, bigraphs. And later by Sassone and Sobociński [21]: their labelled relation largely coincides with ours. The main difference, besides the use of our flavour of C/E nets, is the introduction of saturated bisimilarity and the corresponding characterisation by semi-saturated bisimilarity. Moreover, our equivalence is weak (the number

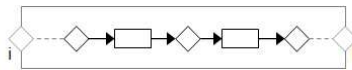
of transitions is not observed) and interleaving (parallelism is reduced to non-determinism). To the best of our knowledge, the treatment of such a bisimilarity for nets, and its decidable characterisation, is original to our paper, and it exploits the results by the second author, König and Montanari reported in [4].

## 5.2 Nets for service equivalence

The application of Petri nets to the specification and the modelling of distributed systems has been around since their inception. Concerning Web services, the use of nets has been strongly advocated in the works by van der Aalst, see e.g. [24] and the position paper [22]. More specifically, he and his coauthors address the issue of equivalences for nets in [23]: they propose a trace-based, probabilistic equivalence for nets which is quite far from our proposal.

Tightly related to us is the work of Martens, which actually inspired some of our examples in Section 3. More precisely, we refer to [11], where van der Aalst’s WF Petri nets (with disjoint sets of input, output and internal places) are analysed for checking structural properties of Web services. The author introduces there the notions of net module and net composition: these recall the open nets formalism mentioned above, and roughly coincide with our own solution.

Martens introduces in [11] the notion of net soundness and net usability, basically related to state reachability with respect to composition with suitable modules. We sketch here a solution for recasting those notions in terms of saturated bisimilarity. Let us start considering the OCPR net **1** depicted in Fig. 5; moreover, let  $\nu_O[-]$  be the OCPR context that close all the elements in the set  $O$  of open data places of an OCPR net  $\mathcal{N}$ . According to [11], we say that a net  $\mathcal{N}$  is *weakly sound* if and only if  $\nu_O[\mathcal{N}] \approx \mathbf{1}$ ; a context  $C[-]$  *utilizes* a net  $\mathcal{N}$  if  $\nu_O[C[\mathcal{N}]] \approx \mathbf{1}$ ; and a net  $\mathcal{N}$  is *usable* if there exists a context  $C[-]$  utilizing it.



**Fig. 5.** The OCPR net **1**.

Next, the author considers there a few notions of observational equivalence for his flavour of WF nets, discussing in turn trace, bisimulation and simulation equivalences. He identifies the weakness of trace equivalence with respect to deadlock, as we echoed in Section 3. He further argues on bisimulation, reaching the conclusion of simulation as the most adequate equivalence. Note that Martens’ notion of simulation (denoted here by  $\equiv$ ) can be expressed by saturated bisimilarity. Indeed,  $\mathcal{N} \equiv \mathcal{N}'$  if and only if  $\nu_O[C[\mathcal{N}]] \approx \mathbf{1} \Leftrightarrow \nu_O[C[\mathcal{N}']] \approx \mathbf{1}$  for all possible contexts  $C[-]$ : intuitively, this means that the two nets are usable by exactly the same environments. Note that our solution is stricter, since  $\approx \subseteq \equiv$ , while the converse does not hold. In fact,  $M_3$  and  $M_4$  of Fig. 4 of [12] are equivalent according to Martens, even if they are not saturated bisimilar.

## 6 Concluding remarks

In our work we first introduced *Open Consume-Produce-Read* (OCPR) nets to naturally model the behaviour of OWL-S Web services (although our approach can be extended to WS-BPEL services as well, in the line of [17]). Next, we defined the notion of *saturated bisimilarity* between two OCPR nets, which relies on the concepts of interface and CPR context. An interface is a set of open (i.e., externally observable) places which interact with the environment, while a CPR context represents a possible environment where a service can be embedded in.

To the best of our knowledge, the proposed bisimilarity is the first equivalence employing Petri nets for Web services that features:

- *weakness* – It abstracts from internal transition steps by equating structurally different, yet externally indistinguishable services. An obvious application of such a notion is for checking whether a (complex) service implements a given specification. Indeed, it is often the case that a service provider publishes simple service specifications by hiding unnecessary and/or confidential details of their implementations, as well as, similarly, a matchmaking system verifies whether a (composition of) service(s) matches a client query.
- *compositionality* – It is the largest bisimulation that is also a congruence. Thus, two equivalent services can be always used interchangeably. Consider, for instance, a complex application where a component service  $S$  fails (e.g., it becomes unavailable).  $S$  can be replaced with an equivalent service  $S'$  without changing the behaviour of the whole application.
- *decidability* – As the set of the states of an OCPR net is finite, the saturated bisimilarity is decidable. It can hence be employed by automated software in order to check whether two services are behaviourally equivalent.

We leave to future work a throughout analysis of the connection between our saturated bisimilarity and Martens' simulation. As for now, we just note that our relying on a standard notion of observational equivalence allows the reuse of existing theoretical techniques and practical tools, such as e.g. the characterizations of minimal equivalent net and the algorithms for calculating it.

## References

1. P. Baldan, A. Corradini, H. Ehrig, and R. Heckel. Compositional semantics for open Petri nets based on deterministic processes. *Mathematical Structures in Computer Science*, 15(1):1–35, 2005.
2. B. Benatallah and R. Hamadi. A Petri net-based model for Web service composition. In K.-D. Schewe and X. Zhou, editors, *Australasian Database Conference, Conferences in Research and Practice in Information Technology 17*, pp. 191–200. Australian Computer Society, 2003.
3. E. Best, R.R. Devillers, and M. Koutny. The Box Algebra = Petri nets + process expressions. *Information and Computation*, 178(1):44–100, 2002.
4. F. Bonchi, B. König, and U. Montanari. Saturated semantics for reactive systems. In *Logic in Computer Science*, pp. 69–80. IEEE Computer Society, 2006.

5. A. Brogi and S. Corfini. Behaviour-aware discovery of Web service compositions. Tech. Rep. TR-06-08, University of Pisa, Department of Computer Science, 2006.
6. OWL-S Coalition. OWL-S 1.1, 2004. <http://www.daml.org/services/owl-s/1.1/>.
7. H. Ehrig and B. König. Deriving bisimulation congruences in the DPO approach to graph rewriting. In I. Walukiewicz, editor, *Foundations of Software Science and Computation Structures, LNCS 2987*, pp. 151–166. Springer, 2004.
8. E. Kindler. A compositional partial order semantics for Petri net components. In P. Azèma and G. Balbo, editors, *Applications and Theory of Petri Nets, LNCS 1248*, pp. 235–252. Springer, 1997.
9. S. Lack and P. Sobociński. Adhesive categories. In I. Walukiewicz, editor, *Foundations of Software Science and Computation Structures, LNCS 2987*, pp. 273–288. Springer, 2004.
10. J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In C. Palamidessi, editor, *Concurrency Theory, LNCS 1877*, pp. 243–258. Springer, 2000.
11. A. Martens. Analyzing Web service based business processes. In M. Cerioli, editor, *Fundamental Approaches to Software Engineering, LNCS 3442*, pp. 19–33. Springer, 2005.
12. A. Martens. Consistency between executable and abstract processes. In *e-Technology, e-Commerce, and e-Services*, pp. 60–67. IEEE Computer Society, 2005.
13. R. Milner. *A Calculus of Communicating Systems, LNCS 92*. Springer, 1980.
14. R. Milner. Bigraphs for Petri nets. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets, LNCS 3098*, pp. 686–701. Springer, 2004.
15. U. Montanari and F. Rossi. Contextual nets. *Acta Informatica*, 32:545–596, 1995.
16. E. Newcomer. *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley, 2002.
17. C. Ouyang, E. Verbeek, W.M.P. van der Aalst, S. Breutel, M. Dumas, and A.H.M. ter Hofstede. Formal semantics and analysis of control flow in WS-BPEL. Tech. Rep. BPM-05-15, BPM Center, 2005.
18. L. Pomello, G. Rozenberg, and C. Simone. A survey of equivalence notions for net based systems. In G. Rozenberg, editor, *Advances in Petri nets, LNCS 609*, pp. 410–472. Springer, 1992.
19. W. Reisig. *Petri Nets: An Introduction*. EACTS Monographs on Theoretical Computer Science. Springer, 1985.
20. V. Sassone and P. Sobociński. Reactive systems over cospans. In *Logic in Computer Science*, pp. 311–320. IEEE Computer Society, 2005.
21. V. Sassone and S. Sobociński. A congruence for Petri nets. In H. Ehrig, J. Padberg, and G. Rozenberg, editors, *Petri Nets and Graph Transformation, ENTCS 127*, pp. 107–120. Elsevier, 2005.
22. W.M.P. van der Aalst. Pi calculus versus Petri nets: Let us eat “humble pie” rather than further inflate the “Pi hype”. *BPTrends*, 3(5):1–11, 2005.
23. W.M.P. van der Aalst, A.K. Alves de Medeiros, and A.J.M.M. Weijters. Process equivalence: Comparing two process models based on observed behavior. In S. Dustdar, J.L. Fiadeiro, and A. Sheth, editors, *Business Process Management, LNCS 4012*, pp. 129–144. Springer, 2006.
24. H.M.W. Verbeek and W.M.P. van der Aalst. Analyzing BPEL processes using Petri nets. In D. Marinescu, editor, *Applications of Petri Nets to Coordination, Workflow and Business Process Management*, pp. 59–78. Florida International University, 2005.
25. W3C. WSDL 1.1, 2001. <http://www.w3.org/TR/wsdl>.