A Net-based Approach to Web Services Publication and Replaceability

Filippo Bonchi†, Antonio Brogi, Sara Corfini, Fabio Gadducci
Department of Computer Science
University of Pisa, Italy
{fibonchi,brogi,corfini,gadducci}@di.unipi.it

Abstract. Web services represent a promising technology for the development of distributed heterogeneous software systems. In this setting, a major issue is to establish whether two services can be used interchangeably in any context. To this aim, our paper first briefly reviews the results contained in a recent article by the same authors, where a suitable notion of behavioural equivalence for Web services was introduced. Our work then extends those results, in order to account for ontology-based service specifications. Next, a concrete example scenario – a car rental system – is presented, and it is then used to illustrate how the equivalence between services can be fruitfully employed for correctly addressing two prominent, modularity-related problems: the publication of correct service specifications and the replaceability of (sub)services.

1. Introduction

Web services are emerging as a promising technology for the development of next generation distributed heterogeneous software systems. Roughly, a Web service is a self-describing software component universally accessible by means of standard protocols (WSDL, UDDI, SOAP). Platform-independence and ubiquity make Web services the building blocks for developing new complex applications [22]. In this scenario, a prominent issue is to establish whether two services are behaviourally equivalent, i.e., such that an external observer can not tell them apart. Yet, standard WSDL interfaces provide services with purely syntactic descriptions: they do not include information on the possible interaction between services, thus inhibiting the a priori verification of any behavioural property.

*Research partly supported by EU FP6-IST IP 16004 SENSORIA and STREP 0333563 SMEPP, and MIUR FIRB TOCAI.It.
†Address for correspondence: Department of Informatics, University of Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy
Various proposals have been recently put forward to feature more expressive service descriptions that include semantics (viz., ontology-based) and behaviour information about services. One of the major efforts in this direction is OWL-S [21], a high-level ontology-based language for describing services, proposed by the OWL-S coalition. 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 behavioural properties of (compositions of) services.

In this perspective we defined in [2] (an extended version appeared as [4]) a suitable notion of behavioural equivalence for OWL-S described Web services represented by means of OCPR nets. OCPR nets (for Open Consume- Produce-Read nets) are a simple variant of the standard Condition/Event Petri nets, designed to address data persistency. In particular, an OCPR net is equipped with two disjoint sets of places, namely, control and data places, to naturally model the control flow and the data flow of a Web service, and with an interface, which establishes those data places that can be observed externally. The main features of the equivalence presented in [2], named saturated bisimilarity, are weakness, as it equates externally indistinguishable services by abstracting from the number of internal steps, and compositionality, as it is also a congruence. Furthermore, the equivalence was proved there to be decidable, by characterizing it in terms of an alternative, clearly decidable behavioural equivalence.

This paper focuses on exploiting the behavioural equivalence introduced in [2, 4] in order to outline a methodology for addressing two specific issues related to service specification. Namely, for checking whether a service specification is equivalent to a service implementation, and whether a (sub)service may replace another (sub)service without altering the behaviour of the whole application. To this end, a typed variant of the OCPR nets proposed in [2, 4] is introduced, in order to possibly equate service specifications whose data may belong to different types in a shared ontology.

The methodology is presented by directly instantiating it on two simple example scenarios, based on a car rental service. In the first scenario we present, we detail the complete behaviour of the car rental service, and we propose a possible specification to externally publish it. We employ bisimilarity to check whether the proposed specification properly describes the concrete service implementation, i.e., to verify whether the externally observable behaviour of the service implementation and of the service specification are equivalent. In the second scenario, we consider the car rental service as part of the package offered by a travel agency service, and we show how it can be replaced by an external, less specific vehicle rental service. By applying bisimilarity, we verify whether these two latter services are equivalent as well as whether the vehicle rental can replace the car rental service affecting neither the internal behaviour of the travel agency service nor its public interface.

The rest of the paper is organized as follows. Section 2 offers a brief review of OWL-S, while Section 3 illustrates the car rental case study. The first two parts of Section 4 briefly recall the results of [4] by discussing OCPR nets and contexts; while the third part introduces the typed variants of OCPR nets and (compatible) contexts. Finally, in Section 5 the notions of (typed) saturated and weak bisimilarity are introduced, and proved to coincide. Section 6 presents a formal encoding of OWL-S service descriptions into typed OCPR nets. Section 7 exploits these results on compositional specification for outlining a methodology addressing the issues on service specification mentioned above. In Section 8 the methodology is then instantiated on the car rental case study described in Section 3. Finally, we discuss related work and we draw some concluding remarks in Section 9.

The paper extends the work presented in [3]. In particular, the introduction of typed OCPR nets is novel, and this impacts the formal service encoding as well as the methodology for addressing service publication and replaceability, discussed in that work. Thus, Section 6 and Section 7 are also basically
original. Furthermore, besides offering a proof of the main result, the appendix presents a precise correspondence between the decidable bisimilarity we introduced in [4] and the standard notion of weak bisimilarity, as it can be found in process calculi literature [19].

2. A Brief Introduction to OWL-S

As anticipated in the Introduction, we consider Web services specified in OWL-S. Quoting the OWL-S coalition [21]: “OWL-S is a OWL-based Web service ontology, which supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form”.

The OWL-S ontology has three primary sub-ontologies: the service profile, the process model and the grounding, each of them modelling a different view of a service. The service profile describes what the service does, the process model describes how the service is used, and the service grounding describe how to interact with the service. Each OWL-S service description consists of one (instance of the class) process model, of one (instance of the class) grounding or more, and, optionally, of one (instance of the class) profile or more. While the process model provides an abstract description of how to interact with the service, the grounding complements the process model providing protocol and message format information to access the service. Multiple profiles may be exposed to provide different high-level service descriptions suitable for service selection.

In the paper we point our attention to the OWL-S process model, as we focus on service behaviour. The process model describes how the service performs its component tasks. 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 (viz., it has no internal structure) and it executes in a single step (similarly to a black box providing a functionality). A composite process consists of a set of component processes and it is built up by using a few control constructs: sequence (i.e., sequential execution), if-then-else (conditional execution), choice (non-deterministic execution), split (parallel execution), split+join (parallel execution with synchronization), any-order (unordered sequential execution), repeat-while and repeat-until (iterative execution).

Let us consider the process model of the sample CarRental service illustrated in Figure 1. Note that Figure 1 presents a more compact, readable tree-view of the CarRental process model, rather than listing the actual (not human-oriented) OWL-S code. Each internal node (viz., a gray node) is labelled with the type of the composite process it represents, and, in case of conditional and iterative control constructs, also with a condition. Leaves (viz., white nodes) represent atomic processes. CarRental is a sequence process whose left-most child is an atomic process (checkCarAvailability) which checks whether the requested car is available in the specified period. If the car is not available (availability = false), CarRental terminates (proposeAlternative) by proposing an alternative rental contract (e.g., a contract for a different car and/or rental period) to the customer. Otherwise (availability = true), CarRental continues by checking whether the given driving licence suffices to drive the requested car. If so (accepted = true), it finalizes the car rental (rentACar), otherwise (accepted = false) it proposes an alternative rental contract (proposeAlternative).

As one may observe in Figure 1, atomic processes have associated inputs and outputs (IOs, for short). Each input (output) is described by a syntactic name and a type. We use the notation name

\[\text{name}_1\text{OWL-S permits to associate atomic processes also with preconditions and results, yet we do not consider them in this work.} \]
: type to represent that the input (output) name has type type. A crucial aspect is that the values of the types of OWL-S inputs and outputs are URIs (Uniform Resource Identifiers) of concepts defined in some ontology. An ontology is a formal representation of a set of concepts and their relationships within a domain. For instance, a fragment of a Vehicle ontology is depicted in Figure 2. Some concepts of the Vehicle domain are, for example, car, bus and truck. Typically, concepts are related by subtype and supertype relationships. For example, car is a (direct) subtype of vehicle (namely, any car is a vehicle, but not every vehicle is a car), as well as sedan is a subtype of vehicle. Dually, vehicle is a supertype of car and sedan. A major advantage of Web-addressable ontology concepts is the possibility of implementing matching processes not failing due to syntactic mismatches of service IO parameters.

Composite processes have associated inputs and outputs, as well. Intuitively, the IOs of a composite process are the set of IOs of its child processes. OWL-S provides a suitable binding mechanism to relate one input (output) of a composite process with one or more inputs (outputs) of its child processes. For the sake of simplicity, IOs of composite processes are not reported in Figure 1. Furthermore, we use the following convention. Let $A_1, A_2, \ldots, A_n$ be the child atomic processes of a composite process $C$. The
inputs (outputs) of $A_1, A_2, \ldots, A_n$ which have the same name, correspond to a single input (output) of $C$. For example, the IOs of the root composite process of CarRental are requestedCar : car, rentPeriod : date, licence : drivingLicence, availability : boolean, altContract : contractProposal, accepted : boolean and contract : contract. Indeed, the inputs requestedCar : car of the atomic processes checkAvailabilityCar and checkDrivingLicence correspond to the single requestedCar : car input of the CarRental root composite process.

3. Case Study: A Car Rental System

In order to provide a proper motivation for our proposals, this section illustrates an example scenario concerning a car rental system. We consider a car rental providing the CarRental service previously introduced in Section 2 and publically advertised by the abstract specification illustrated in Figure 3. As one may note, the public specification of CarRental hides unnecessary and/or confidential details of its implementation (such as the CarRental section which checks the suitability of a given driving licence).

Let us now suppose that the CarRental provider enhances its service by evolving into a vehicle rental system. The process model of the new provided VehicleRental service is illustrated in Figure 4. VehicleRental firstly checks whether the requested vehicle is available in the specified period (checkVehicleAvailability). Note that in this sense VehicleRental is more general than CarRental, since it is in charge of renting vehicles (i.e., not only cars). Similarly to CarRental, if the vehicle is not available ($availability = false$), VehicleRental terminates (proposeAlternative) by proposing an alternative rental contract to the customer. Otherwise ($availability = true$), VehicleRental continues by checking whether the customer holds the driving licence necessary to drive the requested vehicle. If the given licence does not suffice ($accepted = false$) it proposes an alternative rental contract (proposeAlternative). Otherwise ($accepted = true$), VehicleRental determines the level of insurance needed to rent the requested vehicle (computeInsurance), and it finalizes the vehicle rental by providing either a rental contract with a standard insurance (fillContract), or a contract with a high-level insurance (fillFinerContract). Note that in this case VehicleRental is more specific than CarRental, since it provides more detailed rental contracts (i.e., the output concepts standardContract and finerContract of VehicleRental are subtypes of the contract output concept of CarRental).

---

2 A real car rental service is obviously much more complex than the one considered in this paper. Yet, we removed all the non significant details from the CarRental specification in order to keep the example scenario simple and intuitive.
Figure 4. The OWL-S process model of the VehicleRental service.

The (car/vehicle) rental provider needs to verify whether the new offered VehicleRental service can safely replace the old CarRental service, thus implementing the previously published rental specification and not compromising the behaviour of the CarRental clients.

4. Nets for the Formal Reasoning on Service Behaviour

In [2] we introduced Open Consume-Produce-Read (OCPR) nets – a variant of standard Petri nets [26] – to model the behaviour of (OWL-S described) services. In Section 4.1 and Section 4.2 we recall basic definitions and results on OCPR nets from [2, 4], while in Section 4.3 we introduce their typed variant, which is pivotal in extending our service characterization in order to include ontologies [18].

4.1. (Untyped) Consume-Produce-Read nets

Consume-produce-read nets are a simple extension of standard Petri nets, equipped with two disjoint sets of places: the control (consume-produce) places and the data (produce-read) places. Control and data places respectively model the control and the data flow (in particular the persistency of data) of a service.

Definition 4.1. (CPR net)

A consume-produce-read net (simply, CPR net) $N$ is a five-tuple $(CP_N, DP_N, T_N, CF_N, DF_N)$ where

- $CP_N$ is a finite set of control places,
- $DP_N$ is a finite set of data places (disjoint from $CP_N$),
• $T_N$ is a finite set of transitions,
• $CF_N \subseteq (CP_N \times T_N) \cup (T_N \times CP_N)$ is the control flow relation,
• $DF_N \subseteq (DP_N \times T_N) \cup (T_N \times DP_N)$ is the data flow relation.

A marking $M$ for $N$ is a finite set of places in $P_N = CP_N \cup DP_N$.

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-set and produce-set.

**Def. 4.2. (pre-, post-, read-, and produce-set)**

Given a CPR net $N$, we define for each $t \in T_N$ the sets
\[
\begin{align*}
\diamond t &= \{ s \in CP_N \mid (s,t) \in CF_N \} \\
\cdot t &= \{ s \in DP_N \mid (s,t) \in DF_N \}
\end{align*}
\]
The latter denote respectively the pre-set, post-set, read-set, and produce-set of $t$.

Figure 5 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 Figure 5 reads the data places labelled $I_1, I_2, \ldots, I_n$ (this is represented by a straight line) and produces the data places labelled $O_1, \ldots, O_m$ (this is represented by a pointed arrow). In doing so, the control flow passes from the left-most to the right-most control place.

Figure 5. Modelling atomic operations as CPR net transitions.

A marking of the net $N$ coincides with a subset of its set $P_N$ of places, since each place contains 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 $t$.

**Def. 4.3. (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 $(\diamond t \cup \cdot t) \subseteq M$ and $(M \cap \cdot t) \subseteq \diamond t (M$ enables $t)$, and moreover $M' = (M \setminus \diamond t) \cup t \circ \cup \cdot t$.

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

The enabling condition states that the tokens of the pre-read-set of a transition have to be contained in the marking, and that the marking does not contain any token in the post-set of the transition, unless it is consumed and regenerated (as for C/E nets). Note that data places act instead as sinks, that is, the
occurrence of a token may be checked (read), but the token is never consumed, nor it may disable a transition. This is coherent with our underlying modelling choice with respect to Web services, argued in [8], where the persistency of data is assumed: once it is produced, a data remains always available.

4.2. Open CPR Nets and CPR Contexts

The first step for defining compositionality is to equip nets with an interface.

Definition 4.4. (Open CPR net)
Let $N$ be a CPR net. An interface for $N$ is a triple $\langle i, f, OD \rangle$ such that $i \neq f$ and

- $i$ is a control place (i.e., $i \in CP_N$), the initial place;
- $f$ is a control place (i.e., $f \in CP_N$), the final place; and
- $OD$ is a set of data places (i.e., $OD \subseteq DP_N$), 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 $f \in t^\circ$. An open CPR net $N$ (OCPR for short) is a pair $\langle N, O \rangle$, for $N$ a CPR net and $O$ an outer interface for $N$.

Given an open net $N$, $Op(N)$ denotes the set of open places, which consists of those places occurring in the interface, initial and final places included. Furthermore, the places of $N$ not belonging to $Op(N)$ constitute the closed places.

The graphical notation used to represent OCPR nets can be observed, e.g., in Figure 7. The bounding box of the OCPR net there represents the outer interface of the net: the initial and final control places are going to be used to compose the control of services, and the open data places to share data. Note that, conventionally, we write $i_O$ ($f_O$) to denote the initial (final) control place of the outer interface $O$.

Next, we symmetrically define an inner interface for $N$ as an interface such that there is no transition $t \in T_N$ satisfying either $f \in t^\circ$ or $i \in t^\circ$.

Definition 4.5. (CPR context)
A CPR $k$-ary context $C[-]$ is a $k + 2$-tuple $\langle N, O, I_1, \ldots, I_k \rangle$ such that $N$ is a CPR net, $O$ is an outer interface for $N$, $I_1, \ldots, I_k$ are inner interfaces for $N$, and moreover

- $f_O \not\in \{i_1, \ldots, i_k\}$;
- $i_O \not\in \{f_1, \ldots, f_k\}$;
- $\forall j, l \{i_j, f_j\} \cap \{i_l, f_l\} = \emptyset$, with $j \neq l$.

Contexts represent environments in which services can be plugged-in, i.e., possible ways they can be used by other services. Graphically speaking, as one can note in Figure 6, a context is an open net with a hole, represented by a gray area. The border of the hole denotes the inner interface of the context, while the bounding box is the outer interface. An OCPR net can be inserted in a context if the net outer interface and the context inner interface coincide.

Definition 4.6. (CPR composition)
Let $N_j = \langle N_j, O_j \rangle$ be a family of size $k$ of OCPR nets and $C[-] = \langle N, O, I_1, \ldots, I_k \rangle$ be a $k$-ary CPR context. Then, the composite net $C[N_1, \ldots, N_k]$ is the OCPR net with outer interface $O$ such that
In other words, the disjoint union of the $k+1$ nets is performed, except for those places occurring in $O_1,\ldots,O_k$, which are coalesced: this is denoted by our use of the symbol $\oplus$. Moreover, $O$ becomes the set of open places of the resulting net.\(^3\)

### 4.3. Adding Types to CPR Nets

We turn our attention to a novel version of our nets, by adding *types* to places.

**Definition 4.7. (typed CPR net)**

Let $\mathbb{T}$ be a (partially ordered) set of types. A $\mathbb{T}$-typed CPR net $N$ is a six-tuple $(CP_N, DP_N, T_N, CF_N, DF_N, \sigma_N)$ where

- $(CP_N, DP_N, T_N, CF_N, DF_N)$ is a CPR net, and
- $\sigma_N : DP_N \rightarrow \mathbb{T}$ is the *typing function*.

As for the graphical representation, we adopt the same as for the untyped case, with additionally putting the type near the name of a place, as in $d : t$. Moreover, in the following, we consider a chosen set $\mathbb{T}$ of types. Note that we assume that $\mathbb{T}$ is a partially ordered set, as ontologies are often represented as lattices of concepts ordered by the relation “is a sub concept of”.

Next step is suitably revising the notion of composition, in order to account for typing. We first adapt to our nets the usual definitions of read/write places.

**Definition 4.8. (Possibly Read, Possibly Produced places)**

Let $N$ be a CPR-net and let $d \in DP_N$ be a data place. We say that $d$ is a

- $p$ossibly-read place of $N$ (denoted by $d \in DP_{R,N}$) if there is a transition $t \in T_N$ such that $d \in \blacklozenge t$;
- $p$ossibly-produced place of $N$ (denoted by $d \in DP_{P,N}$) if there is a transition $t \in T_N$ such that $d \in t^*$.

We can then define a compatibility relation for nets and contexts.

**Definition 4.9. (compatible contexts)**

Let $\mathcal{N}_j = (N_j, O_j)$ be a family of size $k$ of $\mathbb{T}$-typed OCPR nets and $C[-] = (N, O, I_1, \ldots, I_k)$ be a $k$-ary $\mathbb{T}$-typed CPR context such that $I_j = O_j$ for $1 \leq j \leq k$. Then, the context $C[-]$ is compatible with the family of net $\mathcal{N}_j$ if for all $j$ and for all $d \in O_j$ the typing of $d$ respects the following

\(^3\)Thus, composition is defined up-to isomorphism, except for the places occurring in the interfaces, whose identity is preserved.
• if \( d \in DP^R_{N_j} \) then \( \sigma_{N_j}(d) \supseteq \sigma_N(d) \), and

• if \( d \in DP^P_{N_j} \) then \( \sigma_{N_j}(d) \subseteq \sigma_N(d) \).

Finally, we can define a typed notion of net composition.

**Definition 4.10. (typed composition)**

Let \( N_j = \langle N_j, O_j \rangle \) be a family of size \( k \) of \( T \)-typed OCPR nets and \( C[-] = \langle N, O, I_1, \ldots, I_k \rangle \) be a \( k \)-ary \( T \)-typed CPR context. If \( C[-] \) is compatible with the family, then the composite net \( C[N_1, \ldots, N_k] \) is defined as for the untyped version, with the additional requirement

\[
\sigma_{C[N_1, \ldots, N_k]}(d) = \begin{cases} 
\sigma_N(d) & \text{if } d \in N, \\
\sigma_{N_j}(d) & \text{if } d \in N_j \setminus N. 
\end{cases}
\]

In other terms, if \( d \) belongs to the context \( C[-] \), then the resulting type is the type assigned by \( C[-] \). The reason is that \( d \) may occur in the outer interface of \( C[-] \), and we do not want to modify it. If \( d \) does not occur in \( C[-] \), then it occurs exactly in one net of the family, since these nets are composed via disjoint union, except for those places belonging to the inner interfaces, hence, also to the context.

### 5. Typed Equivalence for Services

This section introduces suitable equivalence relations over typed nets, identifying those services with the same behaviour under all possible environments they are embedded in. We extend the proposal in [4, 3], in order to account for typing.

#### 5.1. Saturated Bisimilarity for OCPR Nets

Defining a behavioural equivalence for (typed) OCPR nets is far from trivial: the simplest proposal accounts for the firing relation, and it describes the evolution of nets as a whole, without considering their interactions with the environment.

Let \( \mathcal{P} \) be the set of all OCPR nets with markings and let \( \text{Obs}(N, M) = Op(N) \cap M \) be the observation made on the net \( N \) with marking \( M \). Let \( \sigma_{N'} |_{O_{N'}} \) denote the restriction of the typing function to the domain \( O_{N'} \). Moreover, let \( \rightarrow_{N} \) be the reflexive and transitive closure of the firing relation \( \rightarrow \) of the underlying net \( N \) of \( N' \).

**Definition 5.1. (naive bisimulation)**

A symmetric relation \( \mathcal{R} \subseteq \mathcal{P} \times \mathcal{P} \) is a naive bisimulation if whenever \((N, M) \mathcal{R} (N', M')\) then

- \( O_N = O_{N'} \) and \( \sigma_{N'} |_{O_{N'}} = \sigma_N |_{O_{N'}} \),

- \( \text{Obs}(N', M) = \text{Obs}(N', M') \), and

- \( M \rightarrow_{N} M_1 \) implies \( M' \rightarrow_{N'} M'_1 \) & \( (N', M_1) \mathcal{R} (N', M'_1) \).

The union of all naive bisimulations is called naive bisimilarity.
Hence, for any marking, bisimilar nets must have the same set of open places, with corresponding types: the same condition could be stated as \( t_N(d) = t_{N'}(d) \) for all \( d \in O_N(= O_{N'}) \). The equivalence above is “naive” in the sense that it clearly fails to be compositional: see [2, Section 4.2].

**Definition 5.2. (saturated bisimulation)**
A symmetric relation \( R \subseteq P \times P \) is a saturated bisimulation if whenever \((N, M) R (N', M')\) then

- \( O_N = O_{N'} \) and \( \sigma_N |_{O_N} = \sigma_{N'} |_{O_{N'}} \),
- \( \text{Obs}(N, M) = \text{Obs}(N', M') \), and
- \( \forall C[-] : M \rightarrow_{C[N]} M_1 \) implies \( M' \rightarrow_{C[N']} M'_1 \) and \((C[N], M_1) R (C[N'], M'_1)\).

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

**Proposition 5.1.** \( \approx_S \) 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 OCPR nets, hence, between services.

**Definition 5.3. (bisimilar nets)**
Let \( N, N' \) be OCPR nets. They are bisimilar, denoted by \( N \approx N' \), if \((N, \emptyset) \approx_S (N', \emptyset)\).

Note that the above definition implies that \((N, M) \approx_S (N', M)\) for all \( M \) markings over open places. The negative side of \( \approx_S \) is that this equivalence seems hard to be automatically decided, due to the quantification over all possible contexts. The main contribution of [3, 4] is the introduction of a labelled transition system that finitely describes the interactions of a net with the environment. We observe that (a variant of) weak bisimilarity on this finite transition system coincides with saturated bisimilarity, and thus it can be automatically checked. In the following, we recast that result by including typed places.

### 5.2. An Equivalent Decidable Bisimilarity

Saturated bisimulation seems conceptually the right notion, as it is argued in [2]. However, it also seems hard to analyze (or automatically verify), due to the universal quantification over contexts. In this section we introduce (a variant of) weak bisimilarity, based on a simple labelled transition system (LTS) distilled from the firing semantics of an OCPR net.

The introduction of a LTS is inspired to the theory of reactive systems [12]. 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 OCPR nets, the reduction relation is given by \( \rangle \), 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 5.4. (labelled transition system)

Let $\mathcal{N}$ be an OCPR net and $\Lambda = (\{+\} \times P_\mathcal{N}) \cup (\{-\} \times CP_\mathcal{N})$ a set of labels, ranged over by $\lambda$, and $\{\tau\} \cup \Lambda$ a set of actions, ranged over by $\mu$. The transition relation for $\mathcal{N}$ is the relation $R_\mathcal{N}$ inductively generated by the set of inference rules below:

\[
\begin{align*}
o &\in Op(\mathcal{N}) \setminus (M \cup \{f\}) & f &\in M & M &\lambda \rightarrow_\mathcal{N} M \cup \{o\} \\
& & & & M &\rightarrow_\mathcal{N} M \setminus \{f\} \\
& & & & M &\mu \rightarrow_\mathcal{N} M'
\end{align*}
\]

where $M \rightarrow_\mathcal{N} M'$ means $\langle M, \mu, M' \rangle \in R_\mathcal{N}$, and $f$ denotes the final place of $\mathcal{N}$, respectively.

Thus, a context may add tokens in open places, as represented by the transition $\rightarrow_\mathcal{N}$, in order to perform a firing. Similarly, a context may consume tokens from the final place $f$. A context cannot interact with the net in other ways, since the initial place $i$ can be used by the context only as a post condition and 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 needing no additional token from the environment.

The theory of reactive systems ensures that, for a suitable choice of labels, the (strong) bisimilarity on the derived LTS is a congruence [12]. However, often such a bisimilarity does not coincide with the saturated one. In the case at hand, we introduce a notion of weak bisimilarity, abstracting away from the number of steps performed by nets, that indeed coincides with the saturated one.

Hereafter we use $\Rightarrow_\mathcal{N}$ to denote the reflexive and transitive closure of $\rightarrow_\mathcal{N}$.

Definition 5.5. (weak bisimulation)

A symmetric relation $\mathcal{R} \subseteq P \times P$ is a weak bisimulation if whenever $(\mathcal{N}, M) \mathcal{R} (\mathcal{N}', M')$ then

- $O_{\mathcal{N}} = O_{\mathcal{N}'}$, and $\sigma_{\mathcal{N}} |_{O_{\mathcal{N}}} = \sigma_{\mathcal{N}'} |_{O_{\mathcal{N}'}}$,
- $Obs(\mathcal{N}, M) = Obs(\mathcal{N}', M')$,
- $M \rightarrow_\mathcal{N} M_1$ implies $M' \rightarrow_\mathcal{N} M'_1$ & $(\mathcal{N}, M_1) \mathcal{R} (\mathcal{N}', M'_1)$, and
- $M \rightarrow_\mathcal{N} M_1$ implies $M' \Rightarrow_\mathcal{N} M'_1$ & $(\mathcal{N}, M_1) \mathcal{R} (\mathcal{N}', M'_1)$.

The union of all weak bisimulations is called weak bisimilarity ($\approx_W$).

The main result, as reported in [4, 3], is stated below.

Theorem 5.1. $\approx_S = \approx_W$.

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

Corollary 5.1. $\approx_S$ is decidable.

6. A Compositional Encoding for OWL-S

Section 3.3 and Section 4 adapted the notion of open net and net composition. In order to lift also the encoding of services, thus tackling ontologies, we need to further refine the compatibility relation.
6.1. Compatible Typing

In order to encode an OWL-S process into an OCPR net, we need to assign a type to each variable, which is in turn assigned by the ontology mapping holding for the service. Note that different variables, with different types, might be bound together in the process: this is accounted for in our encoding by the use of compatible contexts, which may change the typing through the interface.

We assume that our ontology is a lattice with (binary) operations $\sqcap$, $\sqcup$, and with constants $\top$ and $\bot$.

In the following, if a variable $x$ is either read or written by an atomic service $S$, $\sigma_S(x)$ denotes the type of $x$ that is used by $S$. For each variable $x$ in the OWL-S specification we define

$$\sigma^-(x) = \begin{cases} \top, & \text{if } x \text{ is never read} \\ \bigcap_{S \text{ read } x} \sigma_S(x), & \text{else} \end{cases}$$

$$\sigma^+(x) = \begin{cases} \bot, & \text{if } x \text{ is never written} \\ \bigcup_{S \text{ write } x} \sigma_S(x), & \text{else} \end{cases}$$

Intuitively, $\sigma^-(x)$ is the most specific type for variable $x$ globally needed by the service, while $\sigma^+(x)$ is the most general type for variable $x$ produced by the service.

Definition 6.1. (type compatibility)

A OWL-S service specification is typable if $\sigma^+(x) \sqsubseteq \sigma^-(x)$ for all variables $x$. A compatible typing for the service is an ontology mapping such that $\sigma^+(x) \sqsubseteq \sigma(x) \sqsubseteq \sigma^-(x)$ for all variables $x$.

6.2. Presenting the Encoding

First we define the extension of a context for a set of data places.

Definition 6.2. (context extension)

Let $C[-] = \langle N, O, I_1, \ldots, I_k \rangle$ be a context, $A$ be a set of data places and $\sigma_A : A \to \mathbb{T}$ a typing function agreeing with the typing of $C[-]$ (i.e., $\sigma_A$ and $\sigma_N$ coincide for data places in $DP_N \cap A$). The context extension $C_A[-]$ is the context with net $N^A = (CP_N, DP_N \otimes A, T_N, CF_N, DF_N, \sigma_N \otimes \sigma_A)$, with inner interface $I \cup A$ and outer interface $O \cup A$.

Data places are obtained by disjoint union, except for coalescing those places occurring in either $O$ or $I_1, \ldots, I_k$ (denoted by $\otimes$). The agreement boils down to require that the typings $\sigma_N$ and $\sigma_A$ coincide on those coalesced data places. As an example, consider the context $\text{choice}_A[-1, -2]$ illustrated in Figure 6 for $A = \{A_1, \ldots, A_n\}$. It is the extension of the $\text{choice}[-1, -2]$ context (not depicted here) that just contains four transitions and six control places.

In order to define the encoding, for each OWL-S operator $op$ we define a corresponding (possibly binary) context $op[-]$. Figure 6 illustrates the encoding for all the operators. In particular, note that the contexts depicted in Figure 6 are the extensions $op_A[-]$ of contexts $op[-]$ corresponding to $op$.

Definition 6.3. (retyping operator)

Let $A, B$ be sets of data places and $\sigma : A \cup B \to \mathbb{T}$ be a typing function. The retyping operator $\rho^{A,B}_\sigma$ is the context with no transitions, with control places $\{i, f\}$ (initial and final place, respectively), with data places $A \cup B$ (typed by $\sigma$) having as inner interface $\langle i, f, B \rangle$ and outer interface $\langle i, f, A \rangle$, respectively.
7. Net Bisimulation for Publication and Replaceability

Section 4 provides us with the tools which are needed for addressing the methodology concerning service publication and replaceability discussed in Section 3.

7.1. Hiding and Casting Data Places

**Definition 7.1. (casting and hiding operator)**

Let \( A, B \) be sets of data places and let \( \sigma : \ A \cup B \rightarrow \mathbb{T} \) be a typing function. The retyping operator \( \rho^{A,B}_\sigma \) is called a casting operator if \( A = B \), and denoted by \( \gamma^A_\sigma \); while it is called an hiding operator if \( A \subset B \), and denoted by \( \nu^{A,B}_\sigma \).

Intuitively, \( \gamma^A_\sigma [N] \) is the OCPR net \( N \) where the open places are typed by \( \sigma_A \). This composition is defined only if \( \gamma^A_\sigma \) and \( N \) are compatible (Definition 6.1), i.e., only if \( A = Op(N) \) and for each \( x \in Op(N), \sigma_A(x) \subseteq \sigma_N(x) \) when \( x \) is a p-read place, and \( \sigma_N(x) \subseteq \sigma_A(x) \) when \( x \) is a p-produced place. In order to make lighter the notation, we avoid to write the upmost index \( A \), whenever it is clear.

Similar considerations hold for \( \nu^{A,B}_\sigma [N] \). Again for the sake of readability, in the following we omit the second and the third index of an hiding operator, whenever they are clear from the context. In particular, given an OCPR net \( N \), we write \( \nu^A[\cdot] \) to mean the net \( \nu^{A,B}_\sigma [N] \) where \( B \) is the set of open places of \( N \) and \( \sigma_B \) is the typing function of \( N \) (restricted to \( Op(N) \)).

**Proposition 7.1.** Let \( N \) be an OCPR net, let \( \gamma \) be a casting operator, and let \( C[\cdot] \) be a context such that \( \gamma[\cdot] \) and \( C[\cdot] \) are well-defined. Then also \( C[N] \) is well-defined and \( C[\gamma[\cdot][N]] = C[N] \).
Figure 6. Contexts corresponding to OWL-S operators extended for $A = \{A_1, \ldots, A_n\}$. 
Proof:
Since \( \gamma_\sigma[\mathcal{N}] \) is well-defined, i.e., \( \gamma_\sigma \) and \( \mathcal{N} \) are compatible, it follows that the inner and outer interfaces of \( \gamma_\sigma \) are \( \{i, f, \text{Op}(\mathcal{N})\} \) and moreover that \( \forall x \in \text{Op}(\mathcal{N}), \sigma(x) \subseteq \sigma_N(x) \) if \( x \) is a p-read place, and \( \sigma_N(x) \subseteq \sigma(x) \) if \( x \) is a p-produced place.

Since \( C[-] \) and \( \gamma_\sigma[\mathcal{N}] \) are compatible, the inner interface of \( C[-] \) is \( \{i, f, \text{Op}(\mathcal{N})\} \) and moreover that \( \forall x \in \text{Op}(\mathcal{N}), \sigma_{NC}(x) \subseteq \sigma(x) \) if \( x \) is a p-read place and \( \sigma(x) \subseteq \sigma_{NC}(x) \) if \( x \) is a p-produced place.

From transitivity of \( \subseteq \), it trivially follows that \( \forall x \in \text{Op}(\mathcal{N}), \sigma_{NC}(x) \subseteq \sigma_N(x) \) if \( x \) is a p-read place and \( \sigma_N(x) \subseteq \sigma_{NC}(x) \) if \( x \) is a p-produced place. Summarizing \( C[-] \) is compatible with \( \mathcal{N} \).

Now consider \( C[\mathcal{N}] \) and \( C[\gamma_\sigma[\mathcal{N}]] \). The sets of transitions, the sets of control and data places, and the control flow and data flow relations are exactly the same in these nets, since the casting \( \gamma_\sigma \) only changes the types. Moreover, the typing function of \( C[\mathcal{N}] \) and \( C[\gamma_\sigma[\mathcal{N}]] \) are exactly the same, since by the definition of context composition the data places in the inner interface of \( C[-] \) takes the type of the context \( C[-] \).

The encoding presented above maps an OWL-S service into an OCPR net, where all the data places are open, i.e., they belong to the interface. As we are going to see later, this choice roughly corresponds to an orchestration view of the service, where all available data are known. The proposition below will be of use for those cases where it might be necessary to abstract away from irrelevant/confidential data.

**Proposition 7.2.** Let \( \mathcal{N} \) be a net, \( A \subseteq \text{Op}(\mathcal{N}) \) a set of data places, and \( C[-] \) a context such that \( O_{C[-]} \cap A = \emptyset \). If the composite \( C[\nu^A[\mathcal{N}]] \) is well-defined, then \( \forall \sigma_A : A \rightarrow \mathbb{T}, \nu^A[C_{\sigma_A}[\mathcal{N}]] \approx_S C[\nu^A[\mathcal{N}]] \).

In plain terms, removing the places in \( A \) from the interface of a net \( \mathcal{N} \), and then inserting the resulting net in a context \( C[-] \), has the same effect as inserting \( \mathcal{N} \) in a slightly enlarged context \( C_A[-] \), and later on removing those same places (clearly, the places in \( A \) do not occur already in the interface of \( C[-] \)).

### 7.2. On Service Publication

Let us consider an OWL-S process description \( S \), with \( D_S \) the set of data occurring in \( S \). The associated OCPR nets \( [\|S\|_{\sigma_D_S}] \) gives a faithful, abstract representation of the whole behaviour of the service. To check if a service and its public specification coincide, it would then suffice to simply check the equivalence of the associated nets. However, it might well happen that the service provider does not want to make all the details available to an external customer, and thus wants to hide some of the data places. This is performed by simply providing the set of data places \( X \subseteq D_S \), corresponding to the data occurring in \( S \) that should be hidden, and consider \( \nu^X[\|S\|_{\sigma_D_S}] \). Moreover, the public specification could require more specific types for p-read data and produce more general types for p-produced data. Thus, any net (even a much simpler one) equivalent to \( \gamma_\sigma[\nu^X[\|S\|_{\sigma_D_S}]] \) for some typing \( \sigma \), represents a public specification of the service.

### 7.3. On Service Replaceability

Let us consider an OWL-S process description \( S \) and its public specification \( P \) and suppose that we need to replace a subservice of \( S \), called \( R \), with a new service \( T \). We must verify that, after the replacement, the external behaviour of the overall system remains the same.
Let $D_S$, $D_P$, $D_T$ and $D_R$ be the sets of data occurring in, respectively, the descriptions $S$, $P$, $T$ and $R$ and let $\sigma_S$, $\sigma_P$ and $\sigma_T$ be their typing function. For the subservice $R$, we consider the typing function $\sigma_R$ that is the restriction of $\sigma_S$ to $D_T$. Formally, “being $R$ a subservice of $S$” means that $D_R \subseteq D_S$ and that there exists a context $C[\cdot]$ such that $C[[R]]_{\sigma_{DR}} = [[S]]_{\sigma_{DS}}$. Since $\approx_S$ is a congruence, it would then suffice to check that $\|R\|_{\sigma_{DR}} \approx_S \|T\|_{\sigma_{DT}}$ in order to be sure that $R$ and $T$ are interchangeable.

However, this condition is too restrictive, since it would imply that $D_R = D_T$ and $\sigma_{DR} = \sigma_{DT}$. Suppose instead that $T$ produces some data that neither $R$ nor $S$ produce. Or, vice versa, suppose that $R$ produces more data than $T$, but these additional data are not used by the rest of $S$. So, even if (the encodings of) $R$ and $T$ are not bisimilar, replacing $R$ with $T$ does not modify the external behaviour of the overall system, so these two services should still be considered interchangeable.

In order to get a general condition for replaceability, take $Y$ as the subset of $D_R$ containing those data neither in $D_P$ nor used by the rest of the service $S$: formally, “being $Y$ not used by $S$” amounts to say that there exists an OCPR context $C[\cdot]$ such that $\nu_Y[[S]]_{\sigma_S} = C[\nu_Y[[R]]_{\sigma_R}]$. Let us assume the existence of a subset $Z$ of data of $T$ such that $D_T \setminus Z = D_R \setminus Y$ and let $\sigma'_R$ be the restriction of $\sigma_R$ to $D_T \setminus Z$. Thus, we say that the replacement is sound (with respect to public specification $P$) if

$$\nu^Y[[R]]_{\sigma_{DR}} \approx_S \gamma_{\sigma'_R}\nu^Z[[T]]_{\sigma_{DT}}$$

The above condition amounts to say that the external behaviour of $S$ does not change. Indeed, for $X = D_S \setminus D_P$ we have $\|P\|_{\sigma_P} \approx_S \gamma_{\sigma_P}\nu^X[[S]]_{\sigma_S}$ (since $P$ is the public specification of $S$), and requiring that $Y \subseteq X$ is not used in $S$ implies

$$\|P\|_{\sigma_P} \approx_S \gamma_{\sigma_P}\nu^X[[S]]_{\sigma_S} = \gamma_{\sigma_P}\nu^X[[S]]_{\sigma_S} = \gamma_{\sigma_P}\nu^X[[S]]_{\sigma_S} \approx_S \gamma_{\sigma_P}\nu^X[[S]]_{\sigma_S}$$

Now, By Proposition 7.1, it follows that

$$\gamma_{\sigma_P}\nu^X[[S]]_{\sigma_S} \approx_S \gamma_{\sigma_P}\nu^X[[S]]_{\sigma_S}$$

So, the replacement is indeed sound. Finally, note that we may safely assume that the data in $Z$ do not occur in $S$, possibly after some renaming, so that $Z \cap (D_S \setminus Y) = \emptyset$. Now let $\sigma' : T \rightarrow \mathbb{T}$ be equal to $\sigma_T$ restricted to $Z$. By Proposition 7.2 we then obtain

$$\gamma_{\sigma_P}\nu^X[[S]]_{\sigma_S} \approx_S \gamma_{\sigma_P}\nu^X[[S]]_{\sigma_S}$$

Notice that $C^Z_{\sigma'_T}[[T]]_{\sigma_{DT}}$ corresponds to the encoding of the process description $S'$, obtained after replacing $R$ in $S$ with $T$. In this encoding the typing is defined as $\sigma_S$ for all the data occurring in $D_S \setminus Y$ (since all these data occurs in the context $C[\cdot]$ with the typing $\sigma_S$), while as $\sigma_T$ for all the data occurring in $Z$ (recall that $Z \cap (D_S \setminus Y) = \emptyset$).

8. Case Study (continued)

We directly instantiate the previously discussed methodology on the rental system scenario introduced in Section 3. Firstly, we verify whether the full behaviour of the CarRental service (Figure 1) actually satisfies the published interface behaviour description of CarRental (Figure 3). We hence translate both the interface behaviour description and the full behaviour of CarRental into OCPR nets, according
to the OWL-S encoding sketched in Section 6. The resulting nets \textit{PUB} and \textit{CAR} are illustrated in Figures 7 and 8, respectively. We note, for example, that the \texttt{altContract : contractProposal} output parameters of the two atomic processes \texttt{proposeAlternative of the CarRental service in Figure 1 have been translated into a single data place in the OCPR net in Figure 8, since they correspond to a single output parameter of the CarRental root process.}\footnote{We also note that the choice of executing either the transition \texttt{proposeAlternative} or the transition \texttt{checkDrivingLicence} in the net of Figure 1 does not depend in any way from the value of the data place \texttt{availability}, as well as the choice of executing either \texttt{proposeAlternative} or \texttt{rentACar} does not depend from the value of the data place \texttt{accepted}. Indeed, our OWL-S encoding (Figure 6) always translates \texttt{if-then-else} constructs into choice contexts.} We finally note that all the data places of the two nets are open. Consequently, if we compare \textit{CAR} and \textit{PUB} with respect to the behavioural equivalence of Subsection 5.2, the two nets have different interfaces and they are hence externally distinguishable. On the other hand (as discussed in Subsection 7.2), if we apply the hiding operator $\nu^A$ to \textit{CAR}, where $A = \{\texttt{requestedCar : car, rentPeriod : date, availability : boolean, altContract : contractProposal, licence : drivingLicence, contract : contract}\}$ (viz., the open data places of \textit{PUB}), the two nets \textit{PUB} and \textit{CAR} become indistinguishable.

Figure 7. \textit{PUB}: OCPR net representation of the publically published CarRental specification.

Figure 8. \textit{CAR}: OCPR net representation of the CarRental service.
and CarRental – although structurally different – result to be externally indistinguishable. Thus, the process model in Figure 3 is a correct interface behaviour description for the CarRental service.

We now verify whether the (new offered) VehicleRental service (Figure 4) can safely replace CarRental (Figure 1), thus implementing the published rental specification (Figure 3) and not compromising the behaviour of the CarRental clients. We firstly translate the VehicleRental service into the OCPR net \( \mathcal{V} \mathcal{H} \) illustrated in Figure 9. It is worth noting the typing (Def. 6.1) of the VehicleRental specification. Consider the output parameters contract : standardContract and contract : finerContract of the atomic processes fillContract and fillFinerContract, respectively, of the VehicleRental service in Figure 4. They have the same name (viz., contract), hence, according to the convention discussed in Section 2, they correspond to a single output parameter of the VehicleRental root process. The net \( \mathcal{V} \mathcal{H} \) in Figure 9 indeed contains a single contract data place, that however is typed with the ontology concept contract. This is why we consider an ontology where the contract concept is a supertype of both the standardContract and finerContract concepts.

Clearly, the nets \( \mathcal{C} \mathcal{A} \mathcal{R} \) and \( \mathcal{V} \mathcal{H} \) are not equivalent, since they expose different interfaces and are obviously externally distinguishable. However the VehicleRental service can safely replace the CarRental service, not affecting the published rental specification. Following the methodology sketched in Section 7, we take the set \( Y \) of data places of \( \mathcal{C} \mathcal{A} \mathcal{R} \) that do not occur in the published rental specification, in our example \( Y = \{ \text{accepted} : \text{boolean} \} \). Then we take the set \( Z \) of data places which occur in \( \mathcal{V} \mathcal{H} \) and not in \( \nu^Y [\mathcal{C} \mathcal{A} \mathcal{R}] \), thus \( Z = \{ \text{accepted} : \text{boolean}, \text{insuranceLevel} : \text{insurance} \} \), and we apply a (re)typing function \( \gamma_t \) that casts the type of the requestedVehicle data place from Vehicle to Car. Indeed, the net \( \mathcal{V} \mathcal{H} \) is compatible (Def. 4.9) with the (empty) context with outer interface \( O_{\nu^Y [\mathcal{C} \mathcal{A} \mathcal{R}]} \), since the read-only place requestedVehicle : vehicle of \( \mathcal{V} \mathcal{H} \) features a more general type than the data place requestedCar : car of the context. At this point, we just need to check that \( \nu^Y [\mathcal{C} \mathcal{A} \mathcal{R}] \approx_S \gamma_t [\nu^Z [\mathcal{V} \mathcal{H}]] \). Since the equivalence holds, VehicleRental can safely replace the CarRental service.
9. Concluding Remarks

This paper outlines a methodology for addressing two pivotal issues in Service-Oriented Computing: publication of correct service specifications and replaceability of (sub)services. Given (the OWL-S process models of) a service $S_1$, its (verified correct) public specification and a service $S_2$, we want to check whether replacing a sub-component of $S_1$ with $S_2$ does not change the behaviour described in the specification. We thus translate the sub-component of $S_1$ and $S_2$ into typed OCPR nets and we check whether such nets are equivalent by closing those data places that do not occur in the public specification of $S_1$ and by possibly casting the remaining data places. The key ingredients of the methodology are a compositional notion of saturated bisimilarity, its characterization via a weak and decidable bisimulation equivalence, a formal encoding from OWL-S to typed CPR nets, and the definition of hiding and casting operators. The untyped variant of the bisimilarities were introduced in [2, 4], while the hiding operator was proposed in [3]; the casting is not necessary for untyped nets. The work is presented through an example scenario, modelling a car rental service as part of a travel agency specification.

Many approaches using Petri nets to model Web services appear in literature. We discussed the issue in [4], where, in particular, we highlighted the connection of OCPR nets to the workflow nets [28, 1], and we pointed out the correspondence with the notion of simulation introduced by Martens in [15, 16].

In the emerging world of Service-Oriented Computing – where applications are built by combining existing services – the issue of service replaceability gained a prominent role, and new approaches are often introduced. The discussion below briefly sums up some recent proposals that we are aware of.

A logic-based approach for service replaceability has been recently presented in [23], where a context-specific definition of service equivalence is introduced. According to [23], given a $\mu$-calculus formula $\phi$ describing some property, a service $S$, taking part in a specific context $C[-]$, can be replaced by a service $T$ if $\phi$ holds also in $C[T]$. Intuitively, such a notion of context-specific replaceability relaxes the requirements imposed by a notion of service (bi)simulation like [2].

Another relaxed replaceability relation on services is induced by the definition of interaction soundness presented in [25]. Given an environment $E$, a service $S$ in an orchestration $O[S]$ can be replaced by $T$ if the interaction of $O[T]$ and $E$ is lazy sound, that is, if the final node of the graph which represents the interaction of $O[T]$ and $E$ can be reached from every initial node.

Although not presented in term of replaceability, the notion of operating guidelines, introduced in [17, 14] and employed in [13] to formally analyze the interactional behaviour of BPEL processes, also implicitly induces a replaceability relation on services — yet not compositional. An operating guideline is an automaton that concisely represents all the partners that properly interact with a service. A service $S$ interacting with $C$ can be replaced with a service $T$ if $T$ belongs to the operating guidelines of $C$.

A theory for checking the compatibility of service contracts based on a CCS-like calculus is presented in [9, 11]. Using a simple finite syntax (featuring sequencing and external/internal choice constructors) to describe service contracts, they define a notion of preorder on processes (based on must testing) reflecting the ability of successfully interacting with clients. Such a notion induces a replaceability relation that, informally, allows one to replace a service $S_1$ with $S_2$ only if all clients compliant with $S_1$ are also compliant with $S_2$. Such a notion of replaceability is uncomparable with ours, as the former emerges from a synchronous model while the latter emerges from an asynchronous model. It is also worth noting that, in particular, [11] shows the existence of the principal dual contract (reminiscent of operating guideline), i.e., the smallest (namely, the most general) service contract that satisfies the client request.
The compatibility of services is also considered in [27]. The authors employ coloured Petri nets to analyse the compatibility of WS-BPEL services and to define – if needed – a message mediator featuring the correct interoperation of two partially compatible services (i.e., services which provide complementary functionalities, however featuring partially different interfaces and interaction behaviour). Although service replaceability can be defined in terms of service compatibility, the approach in [27] is orthogonal with ours, which defines service replaceability on top of a notion of service equivalence.

Other interesting notions of service replaceability were introduced also in [6, 24]. The approach in [6] models service behaviour as a deterministic automaton and defines substitutability with respect to three different notions of compatibility. In particular, context dependent substitutability states that given a service made of two sub-services $S_1$ and $S_2$, $S_1$ can be replaced by $S'_1$ if $S'_1$ is compatible with $S_2$, while context independent substitutability states that a service $S$ can be replaced by a service $S'$ if $S'$ is compatible with all those services which are compatible with $S$ (analogously to [15, 11]). Our notion of substitutability resembles the notion of context independent substitutability (w.r.t. definition of compatibility 1 of [6]) in an asynchronous and non-deterministic setting. The approach in [24] copes with timed business protocols and defines a notion of time-dependent compatibility/replaceability. Let $P_1, P_2$ be timed business protocols. Then, $P_1$ can replace $P_2$ w.r.t. a client protocol $P_C$ if for each timed interaction trace of $P_2$ and $P_C$ there is a corresponding timed interaction trace of $P_1$ and $P_C$. Yet, we do not consider time constraints in our notion of service replaceability.

Furthermore, our relying on the concept of bisimilarity allows us to benefit from the wealth of tools and algorithms developed so far. Indeed, we can check saturated bisimilarity by constructing a finite labelled transition system and then verifying weak bisimilarity there, exploiting e.g. the classical algorithm proposed in [10]. We are currently implementing such a solution. The expected worst-case time complexity can be roughly estimated in $O(S^2)$, where $S$ denotes the number of the markings of an OCPR net. Indeed, given two OCPR nets, the time needed to construct their transition systems is $O(S)$, while the algorithm in [10] takes $O(S^2)$ for checking the weak bisimilarity. We intend however to develop a more efficient algorithm for checking saturated bisimilarity based on normalized minimization [5].

Finally, it is important to note that – for the sake of simplicity – we used a single range of names for identifying the parameters of the presented services, so that the mapping among parameters of different services is obvious. This is unrealistic, since it is often the case that different services may employ different parameter names. Nevertheless, differently from our previous proposals (as reported in [2, 3, 4]), we annotate each functional parameter with a concept defined in a shared ontology, and we just require the parameter matching to be type compatible. Besides adding flexibility, this looseness may allow for (semi-)automatically determine the mapping between parameters of separate services by employing suitable tools for crossing ontologies. Otherwise, in the case of WS-BPEL [7], for example, such a mapping has to be provided manually. In this perspective our approach can be easily extended to WS-BPEL services, exploiting, e.g., a translation from BPEL processes to workflow nets in [20].

References


Appendix: On Weak Bisimilarity

The reader should be aware that our Definition 5.5 does not correspond to the standard notion of weak bisimilarity, which is reported below, for $\rightarrow_N$ denoting the composite relation $\rightarrow_P \rightarrow_N \rightarrow_P$.

**Definition 9.1. (standard weak bisimulation)**
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a standard weak bisimulation if whenever $(\mathcal{N}, M) \mathcal{R} (\mathcal{N}', M')$ then

- $O_{\mathcal{N}} = O_{\mathcal{N}'}$, and $\sigma_{\mathcal{N}} | O_{\mathcal{N}} = \sigma_{\mathcal{N}'} | O_{\mathcal{N}'}$,
- $\text{Obs}(\mathcal{N}, M) = \text{Obs}(\mathcal{N}', M')$,
- $M \overset{\mu}{\rightarrow}_\mathcal{N} M_1$ implies $M' \overset{\mu}{\rightarrow}_\mathcal{N} M'_1$

The union of all standard weak bisimulations is called standard weak bisimilarity ($\approx_{SW}$).

Clearly, our $\approx_W$ is in general finer than the standard $\approx_{SW}$. For example, the CCS-like terms $a$ and $\tau.a$ are not weakly bisimilar, according to $\approx_W$, while they are so, according to $\approx_{SW}$. Furthermore, $\approx_W$ is easier to verify, since each visible action $\lambda$ must be perfectly matched.

However, our abuse of terminology is justified by Theorem 9.1, stating that the two notions coincide, for CPR nets. We first need a technical lemma.

**Lemma 9.1. (preserving reaction)**
Let $(\mathcal{N}, M)$ be an OCPR net with marking, and let $o \in \text{Op}(\mathcal{N}) \setminus \{f\}$, for $f$ final control place of (the outer interface of) $\mathcal{N}$. Then

- if $M \overset{o}{\rightarrow}_\mathcal{N} N$ and $o \notin M$, then $M \overset{o}{\rightarrow}_\mathcal{N} M' \overset{f}{\rightarrow}_\mathcal{N} N$;
- if $M \overset{f}{\rightarrow}_\mathcal{N} N$ and $f \in M$, then $M \overset{f}{\rightarrow}_\mathcal{N} M' \overset{f}{\rightarrow}_\mathcal{N} N$;

The property is obvious if $o$ is a data place; if $o$ is actually the initial control place of the outer interface of $\mathcal{N}$, the result holds since no internal transition may add a token to the initial place. Similarly, it holds for $f$ since no internal transition may remove a token from the final place.

We may now prove a congruence theorem. For the sake of readability, we let $f$ denote the final control place of the outer interface of the OCPR net at hand.5

**Proposition 9.1. ($\approx_{SW}$ is a congruence with respect to markings)**
Let $(\mathcal{N}, M)$ and $(\mathcal{N}', M')$ be OCPR nets with markings, such that $O_{\mathcal{N}} = O_{\mathcal{N}'}$ and $\sigma_{\mathcal{N}} | O_{\mathcal{N}} = \sigma_{\mathcal{N}'} | O_{\mathcal{N}'}$, and let $U \subseteq \text{Op}(\mathcal{N}) \setminus \{f\}$. If $(\mathcal{N}, M) \approx_{SW} (\mathcal{N}', M')$, then $(\mathcal{N}, M \cup U) \approx_{SW} (\mathcal{N}', M' \cup U)$ and $(\mathcal{N}, M \setminus \{f\}) \approx_{SW} (\mathcal{N}', M' \setminus \{f\})$.

**Proof:**
Let us consider the relations $\mathcal{R}_1 = \{(\mathcal{N}, S + U), (\mathcal{N}', T + U) \text{ such that } (\mathcal{N}, S) \approx_{SW} (\mathcal{N}', T)\}$ and $\mathcal{R}_2 = \{(\mathcal{N}, S + U - f), (\mathcal{N}', T + U - f) \text{ such that } (\mathcal{N}, S) \approx_{SW} (\mathcal{N}', T) \text{ and } f \in S\}$. Now, let $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$. We have to prove that $\mathcal{R}$ is a standard weak bisimulation.

5Additionally, in the proofs $M \setminus \{f\}$, $M \cup U$ and $M \cup \{o\}$ are denoted as $M - f$, $M + U$ and $M + o$, respectively.
We just show that $\mathcal{R}_1$ is so, since the proof for $\mathcal{R}_2$ is analogous.

So, let us consider a marking $U = \{u_1, u_2, \ldots u_n\}$, such that $M \cap U = \emptyset$. If $M + U \overset{+\sigma}{\to} N M + U + o$, then $M \overset{+u_1}{\to} N M + u_1 \overset{+u_2}{\to} N \ldots \overset{+u_n}{\to} N M + U \overset{+\sigma}{\to} N M + U + o$. Since $(N, M) \approx_{SW} (N', M')$, then $M' \overset{+u_1}{\Rightarrow} N' M'_1 \overset{+u_2}{\Rightarrow} N' \ldots \overset{+u_n}{\Rightarrow} N' M'_n \Rightarrow N' M'_{n+1}$ and $(N, M + U + o) \approx_{SW} (N', M'_{n+1})$. Thanks to Lemma 9.1, this means that $M' \overset{+u_1}{\Rightarrow} N' M'_{n+1}$ and $(N, M + U + o) \mathcal{R}_1 (N', M'_{n+1})$.

For transitions labelled with $-f$ and with $\tau$, we can proceed as before. \hfill $\Box$

Now, we can finally prove the correspondence theorem.

**Theorem 9.1.** $\approx_{SW} = \approx_W$.

**Proof:**
First of all notice that $\approx_W \subseteq \approx_{SW}$ since the matching condition of standard weak bisimulation is less requiring than that of our weak bisimulation.

In order to prove that $\approx_{SW} \subseteq \approx_W$ we will prove that the following relation is a weak bisimulation.

$$\{ (N, M), (N', M') \text{ such that } (N, M) \approx_{SW} (N', M') \}$$

First of all notice that $Obs(N, M) = Obs(N', M')$, $O_N = O_{N'}$ and $\sigma_N |_{O_N} = \sigma_{N'} |_{O_{N'}}$ by Definition 9.1. Now, let us check the dynamic part.

If $M \overset{+\sigma}{\to} N M + o$, then $o$ is an open place of $N$ and it is not occurring in $M$. Since $O_N = O_{N'}$ and $Obs(N, M) = Obs(N', M')$, then $o$ is also an open place of $N'$ that is not occurring in $M'$. Thus by Definition 5.4, we also have that $M' \overset{+\sigma}{\Rightarrow} N' M' + o$. Since $\approx_{SW}$ is a congruence with respect to the addition of tokens (Proposition 9.1), then we also have that $(N, M + o) \approx_{SW} (N', M' + o)$.

We can proceed analogously for the transitions labelled with $-f$.

For the $\tau$ transitions the definition of $\approx_{SW}$ and $\approx_W$ are exactly the same. \hfill $\Box$

**Appendix: Proof of Theorem 5.1**

In this section we prove Theorem 5.1 by relying on [4, Theorem 1] for untyped OCPR nets. The latter theorem states that untyped saturated bisimilarity (hereafter denoted by $\approx_S$) and untyped weak bisimilarity (hereafter denoted by $\approx_{SW}$) coincide.

Given a typed OCPR net $\overline{N} = \langle N, O \rangle$, we use $\overline{N} = \overline{(N, O)}$ to denote the corresponding untyped net. Since the operational semantics of typed and untyped nets is defined exactly in the same way, the following property holds.

**Lemma 9.2.** Let $N$ be a typed OCPR net and $\overline{N}$ the corresponding untyped net. For all $M$ markings on $\overline{N}$, we have that $M \rightarrow_N M_1$ if and only if $M \rightarrow_{\overline{N}} M_1$.

**Proposition 9.2.** Let $N$ and $N'$ be typed OCPR nets such that $O_N = O_{N'}$ and $\sigma_N |_{O_N} = \sigma_{N'} |_{O_{N'}}$. Let $M$ and $M'$ be markings on them. Then, $(N, M) \approx_S (N', M')$ if and only if $(\overline{N}, \overline{M}) \approx_{\overline{S}} (\overline{N'}, \overline{M'})$. 

\[ F . Bonchi et al. / A Net-based Approach to Web Services Publication and Replaceability \]
Proof:
We prove that \( (N, M) \approx_S (N', M') \) implies \( (N, M) \approx_S (N', M') \). The other direction is easier.

We will prove that the following relation is a saturated bisimulation.

\[
\mathcal{R} = \{ (N, M), (N', M') \text{ such that } (N, M) \approx_S (N', M') \}
\]

First of all, notice that by hypothesis \( N \) and \( N' \) have the same typing function on open places. Moreover, since \( (N, M) \approx_S (N', M') \), we have that \( \text{Obs}(N, M) = \text{Obs}(N', M') \).

Now suppose that for some context \( C[-], M \rightarrow_{C[N]} M_1 \). Since \( N \) and \( N' \) have the same typing function on open places, then \( C[-] \) is also compatible with \( N' \). Moreover by Lemma 9.2, we have that \( M \rightarrow_{C[N]} M_1 \) and, since \( (N, M) \approx_S (N', M') \), then \( M' \rightarrow_{C[N']} M'_1 \) and \( (N, M_1) \approx_S (N', M'_1) \).

Again, by Lemma 9.2, \( M' \rightarrow_{C[N']} M'_1 \) and \( (C[N], M) \mathcal{R} (C[N'], M') \). \(\Box\)

Since the labelled transition systems of typed and untyped nets coincide, the following property holds.

**Lemma 9.3.** Let \( N \) be a typed OCPR net and \( \overline{N} \) the corresponding untyped net. For all labels \( \mu \) and markings \( M \), we have that \( M \xrightarrow{\mu_N} M_1 \) if and only if \( M \xrightarrow{\mu_{\overline{N}}} M_1 \).

**Proposition 9.3.** Let \( N \) and \( N' \) be typed OCPR nets with the same typing function on open places. Let \( M \) and \( M' \) be markings on them. Then \( (N, M) \approx_W (N', M') \) if and only if \( (\overline{N}, M) \approx_W (\overline{N'}, M') \).

**Proof:**
The proof is analogous to the one of Proposition 9.2. We just use Lemma 9.3 in place of Lemma 9.2. \(\Box\)

Now, by Proposition 9.2 and Proposition 9.3 it follows that \( \approx_W = \approx_S \).