Towards a Unifying Theory for Choreography Conformance and Contract Compliance[*]

Mario Bravetti Gianluigi Zavattaro

Department of Computer Science, University of Bologna, Italy

Abstract. In the context of Service Oriented Computing, contracts are descriptions of the externally observable behaviour of services. Given a group of collaborating services, their contracts can be used to verify whether their composition is sound, i.e., the services are compliant. In this paper, we relate the theory of contracts with the notion of choreography conformance, used to check whether an aggregation of services correctly behaves according to a high level specification of their possible conversations. The main result of this paper is the definition of an effective procedure that can be used to verify whether a service with a given contract can correctly play a specific role within a choreography. This procedure is achieved via composition of choreography projection and contract refinement.

1 Introduction

Service Oriented Computing (SOC) is a novel paradigm for distributed computing based on services intended as autonomous and heterogeneous components that can be published and discovered via standard interface languages and publish/discovery protocols. One of the peculiarities of Service Oriented Computing, distinguishing it from other distributed computing paradigms (such as component based software engineering), is that it is centered around the so-called *message oriented architecture*. This means that, given a set of collaborating services, the current state of their interaction is stored inside the exchanged messages and not only within the services. From a practical viewpoint, this means that it is necessary to include, in the exchanged messages, the so-called correlation information that permits to a service to associate a received message to the correct session of interaction (in fact, the same service could be contemporaneously involved in different sessions at the same time).

Web Services is the most prominent service oriented technology: Web Services publish their interface expressed in WSDL, they are discovered through the UDDI protocol, and they are invoked using SOAP.

Two main approaches for the composition of services are currently under investigation and development inside the SOC research community: service *orchestration* and service *choreography*. According to the first approach, the activities of the composed services are coordinated by a specific component, called the

[*] Research partially funded by EU Integrated Project Sensoria, contract n. 016004.

orchestrator, that is responsible for invoking the composed services and collect their responses. Several languages have been already proposed for programming orchestrators such as XLANG [Tha01], WSFL [Ley01] and WS-BPEL [OAS].

Choreography languages are attracting a lot of attention within W3C, where the most credited choreography language WS-CDL [W3C] is currently under development. Choreographies represent a "more democratic" alternative approach for service composition with respect to orchestrations. Indeed, orchestrations require the implementation of central points of coordination; on the contrary, choreography languages support a high level description of peer-to-peer interactions among services that directly communicate without the mediation of any orchestrator. Unfortunately, choreography languages are not yet popular due to the difficulties encountered while translating the high level description of the composed services into an actual system obtained as combination of autonomous, loosely coupled and heterogeneous components.

As an example of service composition, let us consider a travel agency service that can be invoked by a client in order to reserve both an airplane seat and a hotel room. In order to satisfy the client's request, the travel agency contacts two separate services, one for the airplane reservation and one for the hotel reservation. A choreographic specification of this service composition describes the possible flows of invocations exchanged among the four roles (the client, the travel agency, the airplane reservation service, and the hotel reservation service). A formal specification of a choreography of this kind can be found in the Example 1.

The problem that we consider in this paper can be summarized as follows: given a choreography, we want to define an automatic procedure that can be used to check whether a service correctly plays one of the roles described by the choreography. For instance, given a choreographic specification of the above travel agency example, and an actual travel agency service, we want to check whether the actual service behaves correctly according to the choreographic specification. The solution that we propose to this problem assumes that the services expose in their interface an abstract description of their behaviour. In the service oriented computing literature, this kind of information is referred to as the *service contract* [CL06]. More precisely, the service contract describes the sequence of input/output operations that the service intends to execute within a session of interaction with other services. In particular, we propose to combine choreography projection with service contract refinement. The former permits to extract the expected behaviour of one role and synthesize a corresponding service contract. The latter permits to characterize an entire class of contracts (that refine the contract obtained by projection), for which it is guaranteed that the corresponding services correctly play the considered role in the choreography.

An important property of our theory is that contract refinement is defined locally, i.e., given a correct implementation of a choreography based on the contracts $C_1, \cdots, C_n$, each contract C_i can be replaced by any refinement C_i', and the overall system obtained by composition of $C_1', \cdots, C_n'$ is still a correct implementation. This property permits to retrieve the actual services to be composed to implement the choreography independently one from the other (e.g. contemporaneously querying different service registries) collecting the services that either expose the contract obtained by projection, or one of its refined contracts.

The paper is structured as follows. In Section 2 we introduce our model for choreographies defined in terms of a process calculus. In Section 3 we report the theory of service contracts and refinement. This section is essentially an extension of our previous work [BZ06a]; the main novelty is that in the contract calculus we associate to contracts also an additional information indicating the location where the corresponding service is located. The presence of locations permits to prove new interesting results that did not hold in the theory reported in [BZ06a] (for this reason we need to completely revisit and extend the results already proved in [BZ06a]). Section 4 describes the exploitation of the theory of contract refinement in the context of choreography-based service composition. Finally, Section 5 reports some conclusive remarks and the comparison with the related literature.

2 The Choreography Calculus

Definition 1. (Choreographies) *Let* Operations*, ranged over by* $a, b, c, \cdots$ *and* Roles*, ranged over by* $r, s, t, \cdots$*, be two countable sets of operation and role names, respectively. The set of Choreographies, ranged over by* $H, L, \cdots$ *is defined by the following grammar:*

$$H \quad ::= \quad a_{r \to s} \quad | \quad H + H \quad | \quad H;H \quad | \quad H|H \quad | \quad H^*$$

The invocations $a_{r \to s}$ *means that role* r *invokes the operation* a *provided by the role* s*. The other operators are choice* $\_ + \_$*, sequential* $\_;\_$*, parallel* $\_|\_$*, and repetition* $\_^*$*.*

The operational semantics of choreographies considers two auxiliary terms **1** and **0**. They are used to model the completion of a choreography, which is relevant in the operational modeling of sequential composition. The formal definition is given in Table 1 where we take η to range over the set of labels $\{a_{r \to s} \mid a \in \mathit{Operations}, r, s \in \mathit{Roles}\} \cup \{\surd\}$ (the label $\surd$ denotes completion). The rules in Table 1 are rather standard for process calculi with sequential composition and without synchronization; in fact, parallel composition simply allows for the interleaving of the actions executed by the operands.

Choreographies are especially useful to describe the protocols of interactions within a group of collaborating services. To clarify this point, we present a simple example of a protocol described with our choreography calculus.

Example 1. **(Reservation via Travel Agency)** Let us consider the following choreography composed of four roles: *Client*, *TravelAgency*, *AirCompany* and *Hotel*

$Reservation_{Client \to TravelAgency};$
$(\ (Reserve_{TravelAgency \to AirCompany}; ConfirmFlight_{AirCompany \to TravelAgency})\ |$
$\ (Reserve_{TravelAgency \to Hotel}; ConfirmRoom_{Hotel \to TravelAgency})\ );$
$Confirmation_{TravelAgency \to Client} + Cancellation_{TravelAgency \to Client}$

$$a_{r\rightarrow s} \xrightarrow{a_{r\rightarrow s}} \mathbf{1} \qquad \mathbf{1} \xrightarrow{\surd} \mathbf{0} \qquad H^* \xrightarrow{\surd} \mathbf{0}$$

$$\frac{H \xrightarrow{\eta} H'}{H{+}L \xrightarrow{\eta} H'} \qquad \frac{H \xrightarrow{\eta} H' \quad \eta \neq \surd}{H;L \xrightarrow{\eta} H';L} \qquad \frac{H \xrightarrow{\surd} H' \quad L \xrightarrow{\eta} L'}{H;L \xrightarrow{\eta} L'}$$

$$\frac{H \xrightarrow{\surd} H' \quad L \xrightarrow{\surd} L'}{H|L \xrightarrow{\surd} H'|L'} \qquad \frac{H \xrightarrow{\eta} H' \quad \eta \neq \surd}{H|L \xrightarrow{\eta} H'|L} \qquad \frac{H \xrightarrow{\eta} H' \quad \eta \neq \surd}{H^* \xrightarrow{\eta} H';H^*}$$

Table 1. Semantic rules for contracts (symmetric rules omitted).

According to this choreography, the *Client* initially sends a reservation request to a travel agency, that subsequently contacts in parallel an airplane company *AirCompany* and a room reservation service *Hotel* in order to reserve both the travel and the staying of the client. Then, the travel agency either confirms or cancels the reservation request of the client.

Even if choreography languages represent a simple and intuitive approach for the description of the message exchange among services, they are not yet very popular in the context of service oriented computing. The main problem to their diffusion is that it is not trivial to relate the high level choreography description with the actual implementation of the specified system realised as composition of services that are usually loosely coupled, independently developed by different companies, and autonomous. More precisely, the difficult task is, given a choreography, to lookup available services that, once combined, are ensured to behave according to the given choreography.

In order to formally investigate this problem, we need also a calculus for the description of the behaviour of services. This calculus is reported in the next section; in Section 4 we will formalize a procedure to verify whether a given service can play a specific role within a given choreography.

3 The Theory of Contracts with Locations

We assume a denumerable set of action names $\mathcal{N}$, ranged over by $a, b, c, \ldots$. The set $\mathcal{N}_{con} = \{a_* \mid a \in \mathcal{N}\}$ is the set of contract action names. Moreover, we consider a denumerable set *Loc* of location names, ranged over by $l, l', l_1, \cdots$. The set $\mathcal{N}_{loc} = \{a_l \mid a \in \mathcal{N}, l \in Loc\}$ is the set of located action names. The set $\mathcal{A}_{con} = \mathcal{N}_{con} \cup \{\overline{a}_* \mid a_* \in \mathcal{N}_{con}\}$ is the set of input and output contract actions. The set $\mathcal{A}_{loc} = \mathcal{N}_{loc} \cup \{\overline{a}_l \mid a_l \in \mathcal{N}_{loc}\}$ is the set of input and output located actions. We use $\tau \notin \mathcal{N}$ to denote an internal (unsynchronizable) computation. Given a set of located action names $I \subset \mathcal{N}_{loc}$, we denote: with $\overline{I} = \{\overline{a}_l \mid a_l \in I\}$ the set of output actions performable on those names and with $I_l = \{a \mid a_l \in I\}$ the set of action names with associated location l.

Definition 2. (Contracts and Systems) *The syntax of contracts is defined by the following grammar*

$$\begin{array}{rcl} C & ::= & \mathbf{0} \quad | \quad \mathbf{1} \quad | \quad \tau \quad | \quad a_* \quad | \quad \tau;\overline{a}_* \quad | \quad a \quad | \quad \tau;\overline{a}_l \quad | \\ & & C;C \quad | \quad C{+}C \quad | \quad C|C \quad | \quad C\backslash M \quad | \quad C^* \end{array}$$

where $M \subseteq \mathcal{N}_{con}$. The set of all contracts C is denoted by $\mathcal{P}_{con}$. In the following we will omit trailing "$\mathbf{1}$" when writing contracts.
The syntax of systems (contract compositions) is defined by the following grammar

$$P \quad ::= \quad [C]_l \quad | \quad P \| P \quad | \quad P \backslash\!\backslash L$$

where $L \subseteq \mathcal{A}_{loc}$. A system P is well-formed if: (i) every contract subterm $[C]_l$ occurs in P at a different location l and (ii) no output action with destination l is syntactically included inside a contract subterm occurring in P at the same location l, i.e. actions $\overline{a}_l$ cannot occur inside a subterm $[C]_l$ of P. The set of all well-formed systems P is denoted by $\mathcal{P}$. In the following we will just consider well-formed systems and, for simplicity, we will call them just systems.

We take α to range over the set of syntactical actions $SAct = \mathcal{A}_{con} \cup \mathcal{N} \cup \{\overline{a}_l \mid a_l \in \mathcal{N}_{loc}\} \cup \{\tau\}$.

The operational semantics of contracts is defined by the rules in Table 2 (plus symmetric rules). The operational semantics of systems is defined by the rules in Table 3 plus symmetric rules. We take β to range over the set of actions executable by contracts and systems, $Act = \mathcal{A}_{con} \cup \mathcal{N} \cup \mathcal{A}_{loc} \cup \{\tau\}$. We take λ to range over the set of transition labels $\mathcal{L} = Act \cup \{\surd\}$, where $\surd$ denotes successful termination.

$$\mathbf{1} \xrightarrow{\surd} \mathbf{0} \qquad\qquad \alpha \xrightarrow{\alpha} \mathbf{1}$$

$$\frac{C \xrightarrow{\lambda} C'}{C{+}D \xrightarrow{\lambda} C'} \qquad \frac{C \xrightarrow{\lambda} C' \quad \lambda \neq \surd}{C;D \xrightarrow{\lambda} C';D} \qquad \frac{C \xrightarrow{\surd} C' \quad D \xrightarrow{\lambda} D'}{C;D \xrightarrow{\lambda} D'}$$

$$\frac{C \xrightarrow{a_*} C' \quad D \xrightarrow{\overline{a}_*} D'}{C|D \xrightarrow{\tau} C'|D'} \qquad \frac{C \xrightarrow{\surd} C' \quad D \xrightarrow{\surd} D'}{C|D \xrightarrow{\surd} C'|D'} \qquad \frac{C \xrightarrow{\lambda} C' \quad \lambda \neq \surd}{C|D \xrightarrow{\lambda} C'|D}$$

$$\frac{C \xrightarrow{\lambda} C' \quad \lambda \notin M \cup \overline{M}}{C\backslash M \xrightarrow{\lambda} C'\backslash M} \qquad C^* \xrightarrow{\surd} \mathbf{0} \qquad \frac{C \xrightarrow{\lambda} C' \quad \lambda \neq \surd}{C^* \xrightarrow{\lambda} C';C^*}$$

Table 2. Semantic rules for contracts (symmetric rules omitted).

$$\frac{C \xrightarrow{a} C'}{[C]_l \xrightarrow{a_l} [C']_l} \qquad \frac{C \xrightarrow{\overline{a}_{l'}} C'}{[C]_l \xrightarrow{\overline{a}_{l'}} [C']_l} \qquad \frac{P \xrightarrow{\lambda} P' \quad \lambda \neq \surd}{P \| Q \xrightarrow{\lambda} P' \| Q}$$

$$\frac{P \xrightarrow{a_l} P' \quad Q \xrightarrow{\overline{a}_l} Q'}{P \| Q \xrightarrow{\tau} P' \| Q'} \qquad \frac{P \xrightarrow{\surd} P' \quad Q \xrightarrow{\surd} Q'}{P \| Q \xrightarrow{\surd} P' \| Q'} \qquad \frac{P \xrightarrow{\lambda} P' \quad \lambda \notin L}{P \backslash\!\backslash L \xrightarrow{\lambda} P' \backslash\!\backslash L}$$

Table 3. Semantic rules for contract compositions (symmetric rules omitted).

In the remainder of the paper we use the following notations: $P \xrightarrow{\lambda}$ to mean that there exists P' such that $P \xrightarrow{\lambda} P'$ and, given a sequence of labels $w = \lambda_1 \lambda_2 \cdots \lambda_{n-1} \lambda_n$ (possibly empty, i.e., $w = \varepsilon$), we use $P \xrightarrow{w} P'$ to denote the sequence of transitions $P \xrightarrow{\lambda_1} P_1 \xrightarrow{\lambda_2} \cdots \xrightarrow{\lambda_{n-1}} P_{n-1} \xrightarrow{\lambda_n} P'$ (in case of $w = \varepsilon$ we have $P' = P$, i.e., $P \xrightarrow{\varepsilon} P$).

The main results reported in this paper are consequences of a property of systems that we call *output persistence*. This property states that once a system decides to execute an output whose location is not included in the system, its actual execution is mandatory in order to successfully complete the execution of the system. In order to formally prove this property we need to formalize two (easy to prove) preliminary lemmata. Given a system $P \in \mathcal{P}$, we use $loc(P)$ to denote the subset of Loc of the locations of contracts syntactically occurring inside P: e.g. $loc([C]_{l_1} \| [C']_{l_2}) = \{l_1, l_2\}$.

Proposition 1. (Output persistence) *Let $P \in \mathcal{P}$ be a system such that $P \xrightarrow{w} P' \xrightarrow{\overline{a}_l}$, with $l \notin loc(P)$. We have that, for every P'' such that $P' \xrightarrow{w'} P''$ and $P'' \xrightarrow{\surd}$, the string w' must include $\overline{a}_l$.*

Note that, when we apply external restriction on outputs $\overline{a}_l$ to a system P such that $l \notin loc(P)$, i.e. we consider $P \backslash\!\backslash \overline{O}$, with $O \subset \mathcal{N}_{loc}$ such that $O_l = \emptyset$ for every $l \in loc(P)$, due to the absence of internal communication of actions of $\overline{O}$ inside the system, we obtain a transition system isomorphic to that of the system $P\{\mathbf{0}/\alpha | \alpha \in \overline{O}\}$, i.e. the syntactical substitution of $\mathbf{0}$ for every syntactical occurrence of "α" such that $\alpha \in \overline{O}$. In the following we will use the abuse of notation "$C \backslash\!\backslash \overline{O}$" to stand for "$C\{\mathbf{0}/\alpha | \alpha \in \overline{O}\}$": this allows us, e.g., to write a term $([C_1]_{l_1} \| [C_2]_{l_2}) \backslash\!\backslash \overline{O}$ in the format considered above as $[C_1 \backslash\!\backslash \overline{O}]_{l_1} \| [C_2 \backslash\!\backslash \overline{O}]_{l_2}$. As far as inputs are concerned, we cannot perform a similar symmetric syntactic input action removal in the case of restriction on inputs $\overline{a}_l$ applied to a general system P such that $l \in loc(P)$. This because internal communication inside the system could make use of the inputs that we are considering for removal. However the property holds if we restrict to a system composed of a single contract. When we apply external restriction on input directly to a contract C, i.e. we consider $[C]_l \backslash\!\backslash I$, with $I = \{a_l \mid a \in M\}$ for some $M \subseteq \mathcal{N}$, we obtain a transition system isomorphic to that of $[C\{\mathbf{0}/\alpha | \alpha \in M\}]_l$. In the following we will use the abuse of notation "$C \setminus\!\!\setminus M$" to stand for "$C\{\mathbf{0}/\alpha | \alpha \in M\}$": this allows us to write the term above simply as $[C \setminus\!\!\setminus M]_l$.

We now define the notion of correct composition of contracts. This notion is the same as in [BZ06a]. Intuitively, a system composed of contracts is correct if all possible computations may guarantee completion; this means that the system is both deadlock and livelock free (there could be an infinite computation, but given any possible prefix of this infinite computation, it can be extended to reach a successfully completed computation).

Definition 3. (Correct contract composition) *A system P is a correct contract composition, denoted $P\!\downarrow$, if for every P' such that $P \xrightarrow{\tau}{}^{*} P'$ there exists P'' such that $P' \xrightarrow{\tau}{}^{*} P'' \xrightarrow{\surd}$.*

3.1 Subcontract pre-order

We are now ready to define the notion of subcontract pre-order. Given a contract $C \in \mathcal{P}_{con}$, we use $oloc(C)$ to denote the subset of Loc of the locations of the destinations of all the output actions occurring inside C.

Definition 4. (Subcontract pre-order) *A pre-order $\leq$ over $\mathcal{P}_{con}$ is a subcontract pre-order if, for any $n \geq 1$, contracts $C_1, \ldots, C_n \in \mathcal{P}_{con}$ and $C'_1, \ldots, C'_n \in \mathcal{P}_{con}$ such that $\forall i.\, C'_i \leq C_i$, and distinguished location names $l_1, \ldots, l_n \in Loc$ such that $\forall i.\, l_i \notin oloc(C_i) \cup oloc(C'_i)$, we have*

$$([C_1]_{l_1} \parallel \ldots \parallel [C_n]_{l_n})\!\downarrow \quad \Rightarrow \quad ([C'_1]_{l_1} \parallel \ldots \parallel [C'_n]_{l_n})\!\downarrow$$

We will prove that there exists a maximal subcontract pre-order family; this is a direct consequence of the output persistence property. In fact, if we consider mixed choice it is easy to prove that there exists no maximal subcontract pre-order family (see [BZ06a]).

We will show that the maximal subcontract pre-order family can be achieved defining a more coarse form of refinement in which, given any system composed of a set of contracts, refinement is applied to one contract only (thus leaving the other unchanged). We call this form of refinement *singular subcontract pre-order.*

Intuitively a pre-order $\leq$ over $\mathcal{P}_{con}$ is a singular subcontract pre-order whenever the correctness of systems is preserved by refining just one of the contracts. More precisely, for any $n \geq 1$, contracts $C_1, \ldots, C_n \in \mathcal{P}_{con}$, $1 \leq i \leq n$,$C'_i \in \mathcal{P}_{con}$ such that $C'_i \leq C_i$, and distinguished location names $l_1, \ldots, l_n \in Loc$ such that $\forall k \neq i.\, l_k \notin oloc(C_k)$ and $l_i \notin oloc(C_i) \cup oloc(C'_i)$, we require

$$([C_1]_{l_1} \parallel \ldots \parallel [C_i]_{l_i} \parallel \ldots \parallel [C_n]_{l_n})\!\downarrow \quad \Rightarrow \quad ([C_1]_{l_1} \parallel \ldots \parallel [C'_i]_{l_i} \parallel \ldots \parallel [C_n]_{l_n})\!\downarrow$$

By exploiting commutativity and associativity of parallel composition we can group the contracts which are not being refined and get the following cleaner definition. We let $\mathcal{P}_{conpar}$ denote the set of systems of the form $[C_1]_{l_1} \| \ldots \| [C_n]_{l_n}$, with $C_i \in \mathcal{P}_{con}$, for all $i \in \{1, \ldots, n\}$.

Definition 5. (Singular subcontract pre-order) *A pre-order $\leq$ over $\mathcal{P}_{con}$ is a singular subcontract pre-order if, for any $C, C' \in \mathcal{P}_{con}$ such that $C' \leq C$, $l \in Loc$ such that $l \notin oloc(C_i) \cup oloc(C'_i)$, $P \in \mathcal{P}_{conpar}$ such that $l \notin loc(P)$ we have $([C]_l \| P) \downarrow$ implies $([C']_l \| P) \downarrow$*

We let $\mathcal{P}_{conpres}$ denote the set of systems of the form $([C_1]_{l_1} \| \ldots \| [C_n]_{l_n}) \backslash\backslash L$, with $C_i \in \mathcal{P}_{con}$ for all $i \in \{1, \ldots, n\}$ and $L \subseteq \mathcal{A}_{loc}$.

Proposition 2. *Let $\leq$ be a singular subcontract pre-order. For any $C, C' \in \mathcal{P}_{con}$ such that $C' \leq C$, $l \in Loc$ such that $l \notin oloc(C_i) \cup oloc(C'_i)$, $P \in \mathcal{P}_{conpres}$ such that $l \notin loc(P)$ we have $([C]_l \| P) \downarrow$ implies $([C']_l \| P) \downarrow$*

Proof. *Supposed $P = ([C_1]_{l_1} \| \ldots \| [C_n]_{l_n}) \backslash\backslash L$, let $I, O \subset \mathcal{N}_{loc}$ be such that $I = \{a_{l_i} \mid a_{l_i} \in L \wedge 1 \leq i \leq n\}$ and $O = \{a_l \mid \overline{a}_l \in L\}$ (in O only outputs on the location l in the hypothesis of the proposition are cosidered). We have that $([C]_l \| (P \backslash\backslash L)) \downarrow \iff ([C]_l \| (P \backslash\backslash I \cup \overline{O})) \downarrow \iff ([C]_l \| (P\{\mathbf{0}/\alpha | \alpha \in \overline{O}\} \backslash\backslash I)) \downarrow \iff ([C]_l \| P') \downarrow$, where P' is obtained from $P'' \equiv P\{\mathbf{0}/\alpha | \alpha \in \overline{O}\}$ as follows. We call $M \in \mathcal{N}$ the (finite) set of action names occurring in C and C'. We consider an arbitrary injective function $rel : M \rightarrow \mathcal{N} - M$ that maps each action name a in M into a fresh name $rel(a)$. For each $a_{l'} \in I$, we do the following: (i) we replace each syntactical occurrence of a inside the unique subterm $[C'']_{l'}$ of P'' with $rel(a)$, and (ii) we replace each syntactical occurrence of $\overline{a}_{l'}$ inside P'' with $\overline{rel(a)}_{l'}$. Since the same chain of "$\iff$" holds for C' (using the same relabeling function "rel"), we have that the result is a direct consequence of the definition of singular subcontract pre-order applied to P'.*

In order to prove the existence of the maximal subcontract pre-order, we will prove that every pre-order that is a subcontract is also a singular subcontract (Theorem 1). Moreover we will show that there exists a maximal singular subcontract and we prove that it also a subcontract (Theorem 2).

Theorem 1. *If a pre-order $\leq$ is a subcontract pre-order then it is also a singular subcontract pre-order.*

Proof. *Suppose that $\leq$ is a subcontract pre-order. Consider $n \geq 1$, $C, C' \in \mathcal{P}_{con}$, $l \in Loc$ and $P \in \mathcal{P}_{conpar}$ such that $l \notin loc(P)$. From $([C]_l \| P) \downarrow$ and $C' \leq C$, we can derive $([C']_l \| P) \downarrow$ by just taking in the definition of subcontract pre-order, $C_1 = C$, $C'_1 = C'$, $C_2 \ldots C_n$ to be such that $P = (C_2 \| \ldots \| C_n)$ and finally C'_i to be C_i for every $i \geq 2$ (since $\leq$ is a pre-order we have $C \leq C$ for every C).*

From the simple structure of their definition we can easily deduce that singular subcontract pre-order families have maximum, i.e. there exists a singular subcontract pre-order includes all the other singular subcontract pre-orders.

Definition 6. (Subcontract relation) *A contract C' is a subcontract of a contract C denoted $C' \preceq C$, if and only if for all $l \in Loc$ such that $l \notin oloc(C_i) \cup oloc(C'_i)$ and $P \in \mathcal{P}_{conpar}$ such that $l \notin loc(P)$ we have $([C]_l \| P) \downarrow$ implies $([C']_l \| P) \downarrow$*

It is trivial to verify that the pre-order $\preceq$ is a singular subcontract pre-order and is the maximum of all the singular subcontract pre-orders.

Moreover the *subcontract relation* is also a subcontract pre-order.

Theorem 2. *The pre-order $\preceq$ is a subcontract pre-order.*

Proof. *Consider $n \geq 1$, contracts $C_1, \ldots, C_n \in \mathcal{P}_{con}$ and $C'_1, \ldots, C'_n \in \mathcal{P}_{con}$ such that $\forall i. C'_i \leq C_i$, and distinguished location names $l_1, \ldots, l_n \in Loc$ such that $\forall i. l_i \notin oloc(C_i) \cup oloc(C'_i)$. For any i we let $P_i = [C_i]_{l_i}$ and $P'_i = [C'_i]_{l_i}$. If $(P_1 \| \ldots \| P_n)\!\downarrow$ we can derive $(P'_1 \| \ldots \| P'_n)\!\downarrow$ in n steps: at the i-th step we replace P_i with P'_i without altering the correctness of the system thanks to the property stated by the definition of subcontract relation $\preceq$.*

This last Theorem proves that the maximal singular subcontract pre-order is also a subcontract pre-order; since we proved that every subcontract pre-order is also a singular subcontract pre-order (see Theorem 1), we can conclude that there exists a maximal subcontract pre-order and it corresponds to "$\preceq$".

3.2 Input-Output Subcontract relation

Definition 7. (Input and Output sets) *Given the contract $C \in \mathcal{P}_{con}$, we define $I(C)$ (resp. $O(C)$) as the subset of $\mathcal{N}$ (resp. $\mathcal{N}_{loc}$) of the potential input (resp. output) actions of C. Formally, we define $I(C)$ as follows ($O(C)$ is defined similarly):*

$$I(\mathbf{0}) = I(\mathbf{1}) = I(\tau) = I(a_*) = I(\tau;\overline{a}_*) = I(\tau;\overline{a}_{loc}) = \emptyset \qquad I(a) = \{a\}$$
$$I(C;C') = I(C+C') = I(C|C') = I(C) \cup I(C') \qquad I(C \setminus M) = I(C^*) = I(C)$$

Note that the set M in $C \setminus M$ does not influence $I(C \setminus M)$ because it contains only local names outside $\mathcal{N}$. Given the system P, we define $I(P)$ (resp. $O(P)$) as the subset of $\mathcal{N}_{loc}$ of the potential input (resp. output) actions of P. Formally, we define $I(P)$ as follows ($O(P)$ is defined similarly):

$$I([C]_l) = \{a_l \mid a \in I(C)\} \qquad I(P \| P') = I(P) \cup I(P') \qquad I(P \setminus\!\!\setminus L) = I(P) - L$$

Note that, given $P = (C_1 \| \ldots \| C_n) \setminus\!\!\setminus I \cup \overline{O} \in \mathcal{P}_{conpres}$, we have $I(P) = (\bigcup_{1 \leq i \leq n} I([C_i]_{l_i})) - I$ and $O(P) = (\bigcup_{1 \leq i \leq n} O([C_i]_{l_i})) - O$. In the following we let $\mathcal{P}_{conpres,I,O}$, with $I, O \subseteq \mathcal{N}_{loc}$, denote the subset of systems of $\mathcal{P}_{conpres}$ such that $I(P) \subseteq I$ and $O(P) \subseteq O$.

Definition 8. (Input-Output Subcontract relation) *A contract C' is a subcontract of a contract C with respect to a set of input located names $I \subseteq \mathcal{N}_{loc}$ and output located names $O \subseteq \mathcal{N}_{loc}$, denoted $C' \preceq_{I,O} C$, if and only if for all $l \in Loc$ such that $l \notin oloc(C_i) \cup oloc(C'_i)$ and $P \in \mathcal{P}_{conpres,I,O}$ such that $l \notin loc(P)$ we have $([C]_l \| P)\!\downarrow$ implies $([C']_l \| P)\!\downarrow$*

Due to Proposition 2, we have $\preceq = \preceq_{\mathcal{N}_{loc},\mathcal{N}_{loc}}$. The following Proposition states an intuitive contravariant property: given $\preceq_{I',O'}$, and the greater sets I and O (i.e. $I' \subseteq I$ and $O' \subseteq O$) we obtain a smaller pre-order $\preceq_{I,O}$ (i.e. $\preceq_{I,O} \subseteq \preceq_{I',O'}$). This follows from the fact that extending the sets of input and output actions means considering a greater set of discriminating contexts.

Proposition 3. *Let $C, C' \in \mathcal{P}_{con}$ be two contracts, $I, I' \subseteq \mathcal{N}_{loc}$ be two sets of input channel names such that $I' \subseteq I$ and $O, O' \subseteq \mathcal{N}_{loc}$ be two sets of output channel names such that $O' \subseteq O$. We have:*

$$C' \preceq_{I,O} C \quad \Rightarrow \quad C' \preceq_{I',O'} C$$

The following Proposition states that a subcontract is still a subcontract even if we restrict its actions in order to consider only the inputs and outputs already available in the supercontract.

Proposition 4. *Let $C, C' \in \mathcal{P}_{con}$ be contracts and $I, O \subseteq \mathcal{N}_{loc}$ be sets of located names. We have*

$$\begin{array}{lcl} C' \preceq_{I,O} C & \Rightarrow & C' \backslash\backslash (I(C') - I(C)) \preceq_{I,O} C \\ C' \preceq_{I,O} C & \Rightarrow & C' \backslash\backslash \overline{(O(C') - O(C))} \preceq_{I,O} C \end{array}$$

Proof. *We discuss the result concerned with restriction of outputs (the proof for the restriction of inputs is symmetrical). Let $C' \preceq_{I,O} C$. Given any $P \in \mathcal{P}_{conpres,I,O}$ such that $([C]_l \| P)\downarrow$, we will show that $([C' \backslash\backslash \overline{(O(C') - O(C))}]_l \parallel P)\downarrow$. We first observe that $([C]_l \parallel P \backslash\backslash (O(C') - O(C)))\downarrow$. Since $C' \preceq_{I,O} C$, we derive $([C']_l \parallel P \backslash\backslash (O(C') - O(C)))\downarrow$.*
As a consequence $([C' \backslash\backslash \overline{(O(C') - O(C))}]_l \parallel P \backslash\backslash (O(C') - O(C)))\downarrow$. We can conclude $([C' \backslash\backslash \overline{(O(C') - O(C))}]_l \parallel P)\downarrow$.

All the results discussed so far do not depend on the output persistence property. The first relevant result depending on this peculiarity is reported in the following Proposition. It states that if we substitute a contract with one of its subcontract, the latter cannot activate outputs that were not included in the potential outputs of the supercontract.

Proposition 5. *Let $C, C' \in \mathcal{P}_{con}$ be contracts and $I, O \subseteq \mathcal{N}_{loc}$ be sets of located names and let $C' \preceq_{I,O} C$. For every $l \in Loc$, $l \notin oloc(C_i) \cup oloc(C'_i)$, and $P \in \mathcal{P}_{conpres,I,O}$, $l \notin loc(P)$, such that $([C]_l \| P)\downarrow$,*

$$([C']_l \| P) \xrightarrow{\tau}{}^{*} ([C'_{der}]_l \| P_{der}) \quad \Rightarrow \quad \begin{cases} \forall\, a_{l'} \in O(C') - O(C).\, C'_{der} \overset{\overline{a}_{l'}}{\not\longrightarrow} \\ \forall\, a \in I(C') - I(C).\, P_{der} \overset{\overline{a}_{l}}{\not\longrightarrow} \end{cases}$$

Proof. *We proceed by contradiction for both statements.*

Concerning the first statement. Suppose that there exist C'_{der}, P_{der} such that $([C']_l \| P) \xrightarrow{\tau}{}^{*} ([C'_{der}]_l \| P_{der})$ and $C'_{der} \xrightarrow{\overline{a}_{l'}}$ for some $a_{l'} \in O(C') - O(C)$. We further suppose (without loss of generality) that such a path is minimal, i.e. no intermediate state $(C'_{der2} \| P_{der2})$ is traversed, such that $C'_{der2} \xrightarrow{\overline{a}_{l'}}$ for some $a_{l'} \in O(C') - O(C)$. This implies that the same path must be performable by $([C']_l \backslash\backslash \overline{(O(C') - O(C))} \parallel P)$, thus reaching the state
$([C'_{der}]_l \backslash\backslash \overline{(O(C') - O(C))} \parallel P_{der})$. However, since in the state C'_{der} of contract C' we have $C'_{der} \xrightarrow{\overline{a}_{l'}}$ for some $a_{l'} \in O(C') - O(C)$ and the execution of $\overline{a}_{l'}$

is disallowed by restriction, due to output persistence, the contract will never be able to reach success (no matter what contracts in P will do). Therefore $([C']_l \backslash\backslash \overline{(O(C') - O(C))} \parallel P) \not\downarrow$ and (due to Proposition 4) we reached a contradiction.

Concerning the second statement. Suppose that there exist C'_{der}, P_{der} such that $([C']_l \| P) \xrightarrow{\tau}{}^{*} ([C'_{der}]_l \| P_{der})$ and $P_{der} \xrightarrow{\overline{a}_l}$ for some $a \in I(C') - I(C)$. We further suppose (without loss of generality) that such a path is minimal, i.e. no intermediate state $(C'_{der2} \| P_{der2})$ is traversed, such that $P_{der2} \xrightarrow{\overline{a}_l}$ for some $a \in I(C') - I(C)$. This implies that the same path must be performable by $([C']_l \parallel P \backslash\backslash \overline{(I([C']_l) - I([C]_l))})$, thus reaching the state $([C'_{der}]_l \parallel P_{der} \backslash\backslash \overline{(I([C']_l) - I([C]_l))})$. However, since in the state P_{der} of system P we have $P_{der} \xrightarrow{\overline{a}_l}$ for some $a \in I(C') - I(C)$ and the execution of $\overline{a}_l$ is disallowed by restriction, due to output persistence, the system will never be able to reach success (no matter what contract C will do).
Therefore $([C']_l \parallel P \backslash\backslash \overline{(I([C']_l) - I([C]_l))}) \not\downarrow$. This implies $([C' \backslash\backslash I(C') - I(C)]_l \parallel P) \not\downarrow$ and (due to Proposition 4) we reached a contradiction.

The following Propositions permit to conclude that the set of potential inputs and outputs of the other contracts in the system is an information that does not influence the subcontract relation.

Proposition 6. *Let $C \in \mathcal{P}_{con}$ be contracts, $O \subseteq \mathcal{N}_{loc}$ be a set of located output names and $I, I' \subseteq \mathcal{N}_{loc}$ be two sets of located input names such that $O(C) \subseteq I, I'$. We have that for every contract $C' \in \mathcal{P}_{con}$,*

$$C' \preceq_{I,O} C \quad \iff \quad C' \preceq_{I',O} C$$

Proof. *Let us suppose $C' \preceq_{I',O} C$ (the other direction is symmetric). Given any $l \in Loc$, $l \notin oloc(C_i) \cup oloc(C'_i)$, and $P \in \mathcal{P}_{conpres,I,O}$,$l \notin loc(P)$, such that $([C]_l \| P) \downarrow$, we will show that $([C']_l \| P) \downarrow$. We first observe that $([C]_l \parallel P \backslash\backslash (I - O(C))) \downarrow$. Since $C' \preceq_{I',O} C$ and $O(C) \subseteq I'$, we derive $([C']_l \parallel P \backslash\backslash (I - O(C))) \downarrow$. Due to Proposition 5 we have that $([C']_l \| P \backslash\backslash (I - O(C)))$ can never reach by τ transitions a state where outputs in $O(C') - O(C)$ are executable by some derivative of C', so we conclude $([C']_l \| P) \downarrow$.*

Proposition 7. *Let $C \in \mathcal{P}_{con}$ be contracts, $O, O' \subseteq \mathcal{N}_{loc}$ be two sets of located output names such that for every $l \in Loc$ we have $I(C) \subseteq O_l, O'_l$, and $I \subseteq \mathcal{N}_{loc}$ be a set of located input names. We have that for every contract $C' \in \mathcal{P}_{con}$,*

$$C' \preceq_{I,O} C \quad \iff \quad C' \preceq_{I,O'} C$$

Proof. *Let us suppose $C' \preceq_{I,O'} C$ (the other direction is symmetric). Given any $l \in Loc$, $l \notin oloc(C_i) \cup oloc(C'_i)$, and $P \in \mathcal{P}_{conpres,I,O}$,$l \notin loc(P)$, such that $([C]_l \| P) \downarrow$, we show that $([C']_l \| P) \downarrow$. We observe that $([C]_l \parallel P \backslash\backslash \overline{(O - I([C]_l))}) \downarrow$. Since $C' \preceq_{I,O'} C$ and $I([C]_l) \subseteq O'$, we derive $([C']_l \parallel P \backslash\backslash \overline{(O - I([C]_l))}) \downarrow$. As a consequence $([C' \backslash\backslash I(C') - I(C)]_l \parallel P \backslash\backslash \overline{(O - I([C]_l))}) \downarrow$ and finally $([C' \backslash\backslash I(C') - I(C)]_l \parallel P) \downarrow$. Due to Proposition 5 we have that $([C' \backslash\backslash I(C') - I(C)]_l \parallel P)$ can never reach by τ transitions a state where outputs in $\overline{I([C']_l) - I([C]_l)}$ are executable by some derivative of P, so we conclude $([C']_l \| P) \downarrow$.*

3.3 Resorting to Should Testing

The remainder of this Section is devoted to the definition of an actual procedure for determining that two contracts are in subcontract relation. This is achieved resorting to the theory of *should-testing* [RV05].

First, we need a preliminary result that is a direct consequence of the fact that $C' \preceq_{\mathcal{N}_{loc}, \bigcup_{l \in Loc} I([C]_l)} C$ if and only if $C' \preceq C$.

Lemma 1. *Let $C, C' \in \mathcal{P}_{con}$ be contracts. We have*

$$C' \backslash\backslash (I(C') - I(C)) \preceq C \quad \Rightarrow \quad C' \preceq C$$

Proof. We will show that the hypothesis yields $C' \preceq_{\mathcal{N}_{loc}, \bigcup_{l \in Loc} I([C]_l)} C$. From this we can derive the result by using Proposition 7. Given any $l \in Loc$, $l \notin oloc(C_i) \cup oloc(C'_i)$, and $P \in \mathcal{P}_{conpres, \mathcal{N}_{loc}, \bigcup_{l \in Loc} I([C]_l)}, l \notin loc(P)$, such that $([C]_l \| P)\!\downarrow$, we will show that $([C']_l \| P)\!\downarrow$. We have $([C' \backslash\backslash I(C') - I(C)]_l \parallel P)\!\downarrow \iff ([C' \backslash\backslash I(C') - I(C)]_l \parallel P \backslash\backslash \overline{I([C']_l) - I([C]_l)}) \downarrow \iff ([C']_l \parallel P \backslash\backslash \overline{I([C']_l) - I([C]_l)}) \downarrow \iff ([C']_l \| P)\!\downarrow$.

Note that the opposite implication trivially holds (by taking $O = \mathcal{N}_{loc}$ and $I = \mathcal{N}_{loc}$ in Proposition 4).

In the following we denote with $\preceq_{test}$ the *should-testing* pre-order defined in [RV05] where we consider the set of actions used by terms as being $\mathcal{L} \cup \{\overline{a} \mid a \in \mathcal{N}\}$ (i.e. we consider located and unlocated input and output actions and $\surd$ is included in the set of actions of terms under testing as any other action). We denote here with $\surd'$ the special action for the success of the test (denoted by $\surd$ in [RV05]). In the following we consider λ to range over $\mathcal{L} \cup \{\overline{a} \mid a \in \mathcal{N}\}$.

In order to resort to the theory defined in [RV05], we define a normal form for contracts of our calculus that corresponds to terms of the language in [RV05]. The normal form of the system P (denoted with $\mathcal{NF}(\mathcal{P})$) is defined as follows, by using the operator $rec_X \theta$ (defined in [RV05]) that represents the value of X in the solution of the minimum fixpoint of the finite set of equations θ,

$$\begin{array}{ll} \mathcal{NF}(\mathcal{P}) = rec_{X_1} \theta & \qquad \text{where } \theta \text{ is the set of equations} \\ \quad X_i = \sum_j \lambda_{i,j} ; X_{der(i,j)} & \end{array}$$

where, assuming to enumerate the states in the labeled transition system of P starting from X_1, each variable X_i corresponds to the i-th state of the labeled transition system of P, $\lambda_{i,j}$ is the label of the j-th outgoing transition from X_i, and $der(i, j)$ is the index of the state reached with the j-th outgoing transition from X_i. We assume empty sums to be equal to $\mathbf{0}$, i.e. if there are no outgoing transitions from X_i, we have $X_i = \mathbf{0}$.

Theorem 3. *Let $C, C' \in \mathcal{P}_{con}$ be two contracts. We have*

$$\mathcal{NF}(C' \backslash\backslash I(C') - I(C)) \preceq_{test} \mathcal{NF}(C) \quad \Rightarrow \quad C' \preceq C$$

Proof. According to the definition of should-testing of [RV05], since

$$\mathcal{NF}(C'\backslash\!\backslash(I(C')-I(C))) \preceq_{test} \mathcal{NF}(C)$$

we have that, for every test t, if $\mathcal{NF}(C)$ **shd** *t, then also $\mathcal{NF}(C'\backslash\!\backslash(I(C')-I(C)))$* **shd** *$t$, where Q* **shd** *t iff*

$$\forall w \in \mathcal{L}^*, Q'. \qquad Q\|_{\mathcal{L}}t \xrightarrow{w} Q' \qquad \Rightarrow \qquad \exists v \in \mathcal{L}^*, Q'' : Q' \xrightarrow{v} Q'' \xrightarrow{\surd'}$$

where $\|_{\mathcal{L}}$ is the CSP parallel operator: in $R\|_{\mathcal{L}}R'$ transitions of R and R' with the same label λ (with $\lambda \neq \tau, \surd'$) are required to synchronize and yield a transition with label λ.

Let us now consider $l \in Loc$, $l \notin oloc(C_i) \cup oloc(C_i')$, and $P \in \mathcal{P}_{conpar}$, $l \notin loc(P)$, such that $([C]_l\|P)\downarrow$. We consider $t = \mathcal{NF}(P)\{\surd/\surd;\surd'\}\{\overline{a}/\overline{a}_l | a \in \mathcal{N}\}$, i.e., the normal form of P where: we replace each occurrence of $\surd$ with the sequence $\surd;\surd'$ and we turn every output action $\overline{a}_l$ directed to $[C]_l$ into $\overline{a}$. We denote with $\overline{t}$ the term obtained by turning each $a(a_l)$ occurring in t into $\overline{a}$ ($\overline{a}_l$), and each $\overline{a}(\overline{a}_l)$ into $a(a_l)$. From the definition of **shd** *it immediately follows that $\mathcal{NF}(C)$* **shd** *$\overline{t}$. Since $\mathcal{NF}(C'\backslash\!\backslash(I(C')-I(C))) \preceq_{test} \mathcal{NF}(C)$, we have that also $\mathcal{NF}(C'\backslash\!\backslash(I(C')-I(C)))$* **shd** *$\overline{t}$. From the definition of* **shd** *we can conclude that $(C'\backslash\!\backslash I(C')-I(C)\|P)\downarrow$. The thesis directly follows from Lemma 1.*

In [BZ06a] you can find counter examples that prove that the opposite implication $C' \preceq C \Rightarrow \mathcal{NF}(C'\backslash\!\backslash I(C')-I(C)) \preceq_{test} \mathcal{NF}(C)$ does not hold in general.

4 Contract-based Choreography Conformance

In this section we discuss how to exploit the choreography and the contract calculus in order to define a procedure that checks whether a service exposing a specific contract C can play the role r within a given choreography.

First of all we need to uniform the choreography and the contract calculus. From a syntactical viewpoint, we have to map the operation names used for choreographies with the names used for contracts assuming $Operations = \mathcal{N}$. We do the same also for the role names that are mapped into the location names, i.e., $Roles = Loc$. From the point of view of the operational semantics, we need to slightly modify the labels in the operational semantics of the contract calculus in order to have labels comparable to those used in the choreography calculus. To this aim we have to add the auxiliary set of labels $\{a_{r\rightarrow s}, \overline{a}_{rs} \mid a \in Operations, r, s, \in Roles\}$ and replace the second and the fourth rules in Table 3 with the following ones:

$$\frac{C \xrightarrow{\overline{a}_s} C'}{[C]_r \xrightarrow{\overline{a}_{rs}} [C']_r} \qquad\qquad \frac{P \xrightarrow{a_s} P' \quad Q \xrightarrow{\overline{a}_{rs}} Q'}{P\|Q \xrightarrow{a_{r\rightarrow s}} P'\|Q'}$$

With $P \xrightarrow{\tau^*} P'$ we denote the existence of a (possibly empty) sequence of τ-labeled transitions starting from the system P and leading to P'. Given the sequence of labels $w = \lambda_1 \cdots \lambda_n$, we write $P \overset{w}{\Longrightarrow} P'$ if there exist $P_1, \cdots, P_m$ such that $P \xrightarrow{\tau^*} P_1 \xrightarrow{\lambda_1} P_2 \xrightarrow{\tau^*} \cdots \xrightarrow{\tau^*} P_{m-1} \xrightarrow{\lambda_n} P_m \xrightarrow{\tau^*} P'$.

We are now ready to formalize the notion of correct implementation of a choreography. Intuitively, a system implements a choreography if it is a correct composition of contracts and all of its conversations (i.e. the possible sequences of message exchanges), are admitted by the choreography.

Definition 9. (Choreography implementation) *Given the choreography H and the system P, we say that P implements H (written $P \propto H$) if*

- *P is a correct contract composition and*
- *given a sequence w of labels of the kind $a_{r \rightarrow s}$, if $P \overset{w\surd}{\Longrightarrow} P'$ then there exists H' such that $H \xrightarrow{w\surd} H'$.*

Note that it is not necessary for an implementation to include all possible conversations admitted by a choreography.

Example 2. **(Implementation of the Travel Agency Choreography)** As an example, we present a possible implementation of the choreography reported in the Example 1.

$$\begin{array}{l}
[\tau; \overline{Reservation}_{TravelAgency}; Confirmation]_{Client} \parallel \\
[Reservation; (\tau; \overline{Reserve}_{AirCompany}; ConfirmFlight \mid \\
\qquad\qquad\qquad \tau; \overline{Hotel}_{AirCompany}; ConfirmRoom); \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \tau; \overline{Confirmation}_{Client}]_{TravelAgency} \parallel \\
[Reserve; \tau; \overline{ConfirmFlight}_{TravelAgency}]_{AirCompany} \parallel \\
[Reserve; \tau; \overline{ConfirmRoom}_{TravelAgency}]_{Hotel}
\end{array}$$

Note that in this implementation we assume that the travel agency always replies positively to the request of the client sending the *Confirmation* message.

We are now in place for the definition of the (family of) relations $C \triangleleft_H r$ indicating whether the contract C can play the role r in the choreography H.

Definition 10. (Conformance family) *Let $\triangleleft_H$ to denote relations between contracts and roles parameterized on the choreography H defined on the roles $r_1, \cdots, r_n$. A family of relations $\{\triangleleft_H \mid H \in Choreographies\}$ is a* conformance family *if we have that if $C_1 \triangleleft_H r_1, \cdots, C_n \triangleleft_H r_n$ then $[C_1]_{r_1} \| \cdots \| [C_n]_{r_n} \propto H$*

It is interesting to observe that, differently from the subcontract pre-order families defined on contracts in the previous Section, there exists no maximal conformance family. For instance, consider the choreography $H = a_{r \rightarrow s} | b_{r \rightarrow s}$. We could have two different conformance families, the first one including $\triangleleft_H^1$ such that

$$(\tau; \overline{a}_s | \tau; \overline{b}_s) \triangleleft_H^1 r \qquad\qquad (\tau; a; b \ + \ \tau; b; a) \triangleleft_H^1 s$$

and the second one including $\triangleleft_H^2$ such that

$$(\tau; \overline{a}_s; \tau; \overline{b}_s + \tau; \overline{b}_s; \tau; \overline{a}_s) \triangleleft_H^2 r \qquad\qquad (a|b) \triangleleft_H^2 s$$

It is easy to see that it is not possible to have a conformance family that comprises the union of the two relations $\lhd^1_H$ and $\lhd^2_H$. In fact, the system

$$[\tau;\overline{a}_s;\tau;\overline{b}_s \;+\; \tau;\overline{b}_s;\tau;\overline{a}_s]_r \parallel [\tau;a;b \;+\; \tau;b;a]_s$$

is not a correct composition because the two contracts may internally select two incompatible orderings for the execution of the two message exchanges (and in this case they stuck).

The remainder of the paper is dedicated to the definition of a mechanism that, exploiting the notion of contract refinement defined in the previous section, permits to effectively characterize an interesting conformance family. The first step of this mechanism requires the definition of the projection of a choreography on a specific role.

Definition 11. (Choreography projection) *Given a choreography H, the projection H on the role r, denoted with $[\![H]\!]_r$, is defined inductively on the syntax of H in such a way that*

$$[\![a_{r\rightarrow s}]\!]_t \;=\; \begin{cases} \tau;\overline{a}_s & \textit{if } t = r \\ a & \textit{if } t = s \\ \mathbf{1} & \textit{otherwise} \end{cases}$$

and that it is a homomorphism with respect to all operators.

It is interesting to observe that given a choreography H, the system obtained composing its projections is not ensured to be an implementation of H. For instance, consider the choreography $a_{r\rightarrow s}\ ;\ b_{t\rightarrow u}$. The system obtained by projection is $[\overline{a}_s]_r \parallel [a]_s \parallel [\overline{b}_u]_t \parallel [b]_u$. Even if this is a correct composition of contracts, it is not an implementation of H because it comprises the conversation $b_{t\rightarrow u}a_{r\rightarrow s}$ which is not admitted by H.

The problem is not in the definition of the projection, but in the fact that the above choreography cannot be implemented preserving the message exchanges specified by the choreography. In fact, in order to guarantee that the communication between t and u is executed after the communication between r and s, it is necessary to add a further message exchange (for instance between s and r) which is not considered in the choreography. This problem has been already investigated in [CHY07] where a notion of well formed choreography is introduced, and it is proved that well formed choreographies admit a correct projection.[1]

Nevertheless, the notion of well formed choreography in [CHY07] is rather restrictive. In particular, after the execution of a message sent from the role v to the role z, the subsequent message in the conversation should be mandatorily emitted by z. For instance, the choreography $a_{r\rightarrow s};b_{r\rightarrow s}$ does not satisfy this constraint even if the system $[\tau;\overline{a}_s;\tau;\overline{b}_s]_r \parallel [a;b]_s$ obtained by projection is a correct implementation.

[1] The projection defined in [CHY07] is more complex than ours as their choreography calculus comprises also an explicit notion of session.

To be less restrictive than [CHY07], we consider as well formed all those choreographies for which the system obtained by projection is ensured to be a correct implementation.

Definition 12. (Well formed choreography) *A choreography H, defined on the roles $r_1, \cdots, r_n$, is* well formed *if* $[\,[\![H]\!]_{r_1}\,]_{r_1} \parallel \cdots \parallel [\,[\![H]\!]_{r_n}\,]_{r_n} \propto H$

It is worthwhile to note that well formedness is decidable. In fact, given a choreography H, it is sufficient to take the corresponding system P obtained by projection, then consider P and H as finite state automata, and finally check whether the language of the first automaton is included in the language of the second one. Note that the terms P and H can be seen as finite state automata thanks to the fact that their infinite behaviours are defined using Kleene-star repetitions instead of general recursion.

Now, we define the notion of *consonance* between contracts and roles of a given choreography, and we prove that it is a conformance family.

Definition 13. (Consonance) *We say that the contract C is* consonant *with the role r of the well formed choreography H (written $C \bowtie_H r$) if*

$$\mathcal{NF}\big(C_r \backslash\!\backslash I([\![H]\!]_r) - I(C)\big) \preceq_{test} \mathcal{NF}\big([\![H]\!]_r\big)$$

where $\backslash\!\backslash$, defined in Section 3, is the restriction operator that acts independently on input and output actions; $I(\_)$, defined in Section 3.2, is the function that extracts from a contract the names used as inputs; $\mathcal{NF}(\_)$, defined in Section 3.3, is the function that returns the normal form of a contract; and $\preceq_{test}$, defined in Section 3.3, is the should-testing pre-order.

Theorem 4. *The family $\{\bowtie_H \;\mid H$ is a well formed choreography$\}$ of consonance relations is a conformance family.*

Proof. Suppose that H is a well formed choreography defined on the roles $r_1, \cdots, r_n$. Given $C_1, \cdots, C_n$ such that $C_1 \bowtie_H r_1, \cdots, C_n \bowtie_H r_n$, we have to prove that

1. *$[C_1]_{r_1} \parallel \cdots \parallel [C_n]_{r_n}$ is a correct composition of contracts and*
2. *given a sequence w of labels of the kind $a_{r \rightarrow s}$, if $[C_1]_{r_1} \parallel \cdots \parallel [C_n]_{r_n} \stackrel{w\surd}{\Longrightarrow} P'$ then also $H \xrightarrow{w\surd} H'$.*

We first consider item 1. As H is a well formed choreography we have that (by Definitions 12 and 9) $[[\![H]\!]_{r_1}]_{r_1} \parallel \cdots \parallel [[\![H]\!]_{r_n}]_{r_n}$ is a correct composition of contracts. By definition of $\bowtie_H$ it follows that

$$\mathcal{NF}\big(C_i \backslash\!\backslash (I([\![H]\!]_{r_i}) - I(C_i))\big) \preceq_{test} \mathcal{NF}\big([\![H]\!]_{r_i}\big)$$

for $i \in 1 \cdots n$. By Theorem 3 we have that $C_{r_i} \preceq [\![H]\!]_{r_i}$ and by Theorem 2 it follows that $[C_1]_{r_1} \parallel \cdots \parallel [C_n]_{r_n}$ is a correct composition of services.

In order to prove that also item 2. holds, we start observing that the should testing pre-order is preserved by parallel composition operator [RV05]. Nevertheless, in [RV05] a different parallel operator in the CSP-style is considered:

$P\|_A Q$ (where A is a set of actions) P and Q synchronize on equal actions (and not complementary actions), and only on the actions appearing in the set A. This kind of parallel composition operator is not associative; as it is more convenient for our purposes, we assume right associativity, i.e. $P\|_A Q\|_B R$ stands for $P\|_A (Q\|_B R)$.

Due to preservation of parallel composition we have that

$$\mathcal{NF}\big(C_i \backslash\backslash (I(\llbracket H \rrbracket_{r_i}) - I(C_i))\big) \preceq_{test} \mathcal{NF}\big(\llbracket H \rrbracket_{r_i}\big)$$

implies that also

$$\begin{array}{l}\mathcal{NF}\big(C_1 \backslash\backslash (I(\llbracket H \rrbracket_{r_1}) - I(C_1))\big)\|_{A_1} \cdots \|_{A_{n-1}} \mathcal{NF}\big(C_n \backslash\backslash (I(\llbracket H \rrbracket_{r_n}) - I(C_n))\big) \preceq_{test} \\ \mathcal{NF}\big(\llbracket H \rrbracket_{r_1}\big)\|_{A_1} \cdots \|_{A_{n-1}} \mathcal{NF}\big(\llbracket H \rrbracket_{r_n}\big)\end{array}$$

for any $A_1 \cdots A_n$. Moreover, by applying the same reasoning reported in the proof of Theorem 3, the restrictions on the additional inputs $I(\llbracket H \rrbracket_{r_i}) - I(C_i))$ are not relevant because the other partners do not emit actions on these operations. Thus we have also that

$$\mathcal{NF}\big(C_1\big)\|_{A_1} \cdots \|_{A_{n-1}} \mathcal{NF}\big(C_n\big) \preceq_{test} \mathcal{NF}\big(\llbracket H \rrbracket_{r_1}\big)\|_{A_1} \cdots \|_{A_{n-1}} \mathcal{NF}\big(\llbracket H \rrbracket_{r_n}\big)$$

We now apply a transformation (that preserves should testing pre-order) on the terms $\mathcal{NF}\big(C_i\big)$ and $\mathcal{NF}\big(\llbracket H \rrbracket_{r_i}\big)$ and define A_i in such a way that we achieve CSP terms that have the same sequences of exchanged messages of the corresponding terms of our contract calculus. The first transformation, denoted with $\mathtt{csp}(D)_r^{s_1 \cdots s_m}$, is applied to a contract D playing the role r, that interacts with the other roles $s_1, \cdots, s_m$, in order to transform input and output actions in the corresponding actions of the kind $a_{r \to s}$:

$$\mathtt{csp}(D)_r^{s_1 \cdots s_m} \;=\; \mathcal{NF}(D)\ \{a_{r \to s_i}/\overline{a}_{s_i}\ ,\ \sum_{t \in s_1 \cdots s_m} a_{t \to r}/a \mid a \in \mathit{Operations}\}$$

We have that

$$\begin{array}{rcl}P &=& \mathcal{NF}\big(\mathtt{csp}(C_1)_{r_1}^{r_2 \cdots r_n}\big)\|_{A_1} \cdots \|_{A_{n-1}} \mathcal{NF}\big(\mathtt{csp}(C_n)_{r_n}^{r_1 \cdots r_{n-1}}\big) \;\preceq_{test} \\ && \mathcal{NF}\big(\mathtt{csp}(\llbracket H \rrbracket_{r_1})_{r_1}^{r_2 \cdots r_n}\big)\|_{A_1} \cdots \|_{A_{n-1}} \mathcal{NF}\big(\mathtt{csp}(\llbracket H \rrbracket_{r_n})_{r_n}^{r_1 \cdots r_{n-1}}\big) \;=\; Q\end{array}$$

We now define A_i in such a way that the sequences of actions of the kind $a_{r \to s}$ of P and Q correspond to those of $[C_1]_{r_1} \| \cdots \| [C_n]_{r_n}$ and $[\llbracket H \rrbracket_{r_1}]_{r_1} \| \cdots \| [\llbracket H \rrbracket_{r_n}]_{r_n}$, respectively. Consider, in fact $A_i = \{a_{r_i \to r_j}, a_{r_j \to r_i} \mid a \in \mathit{Operations}, i < j \leq n\}$. As H is well formed, then $[\llbracket H \rrbracket_{r_1}]_{r_1} \| \cdots \| [\llbracket H \rrbracket_{r_n}]_{r_n}$ is a correct implementation thus its sequences of actions $a_{r \to s}$ are included in those of H. Item 2. simply follows from the fact that the should-testing pre-order implies also trace inclusion, thus the traces of $[C_1]_{r_1} \| \cdots \| [C_n]_{r_n}$ are included in those of $[\llbracket H \rrbracket_{r_1}]_{r_1} \| \cdots \| [\llbracket H \rrbracket_{r_n}]_{r_n}$.

5 Related Work and Conclusion

We have addressed the problem of the deployment of service compositions via choreography specifications in the context of service oriented computing. In particular, we have formalized service choreographies and service contracts via process calculi and, exploiting the notion of choreography projection in combination with service contract refinement, we have defined a new relation called *consonance*. The consonance relation is parameterized on a given choreography H and relates service contracts to roles: if a contract C is consonant to a role r, then the services exposing contract C (or one of its refinements) correctly play role r in the considered choreography H.

Choreography languages have been already investigated in a process algebraic setting by Carbone et al. [CHY07] and by Busi et al. [BGG$^+$05,BGG$^+$06].

The paper [CHY07] is the first one, to the best out knowledge, in which the problem of ill-formed choreographies is considered: a choreography is ill-formed when it is not possible to achieve by projection a correct implementation that preserves the message exchanges specified by the choreography. The solution to this problem presented in [CHY07] is given by three basic principles that, when satisfied by a choreography, ensure to achieve a corresponding correct projection. On the one hand, the calculi proposed in [CHY07] are more expressive than the calculi we define in this paper because they comprise name passing and an explicit notion of session. On the other hand, the basic principles imposed in [CHY07] give rise to a more restrictive notion of well formed choreography with respect to the one proposed in this paper (this technical aspect is discussed in Section 4). In [BGG$^+$05,BGG$^+$06] a more general notion of conformance between a choreography and a corresponding implementation as a service system is defined. According to this more general notion of conformance the implementation does not necessarily follow from projection, but additional services (not included at the choreography level) can be added in order to synchronize the correct scheduling of the the message flow.

The theory of contracts that we discuss in Section 3 is an extension of the theory reported in our paper [BZ06a]. More precisely, Section 3 is a revisitation of that theory in a slightly different context. The main novelty here is that we associate a location to each contract, and we assume that output operations are specified by indicating, besides the name of the invoked operation, also its actual location. This difference has a very important consequence which is proved as an original result in this paper: contract refinement is no longer influenced by the set of output operations that can be executed by the other composed contracts. More precisely, contract refinement was defined in [BZ06a] with an associated parameter (the set of output operations available in the other composed contracts) while in the present paper we can define a new notion of contract refinement independently of this information. The details of this new results are discussed in Section 3.

The notion of contract refinement that we propose is achieved resorting to the theory of testing. There are some relevant differences between our form of testing and the traditional one proposed by De Nicola-Hennessy [DH84]. The most relevant difference is that, besides requiring the success of the test, we impose also that the tested process should successfully complete its execution. This further requirement has important consequences; for instance, we do not distinguish between the always unsuccessful process $\mathbf{0}$ and other processes, such as $a.\mathbf{1} + a.b.\mathbf{1}$,[2] for which there are no guarantees of successful completion in any possible context. Another relevant difference is in the treatment of divergence: we do not follow the traditional catastrophic approach, but the fair approach introduced by the theory of should-testing of Rensink-Vogler [RV05]. In fact, we do not impose that all computations must succeed, but that all computations can always be extended in order to reach success.

Contracts have been investigated also by Fournet et al. [FHR$^+$04] and by Carpineti et al. [CCL$^+$06]. In [FHR$^+$04] contracts are CCS-like processes; a generic process P is defined as compliant to a contract C if, for every tuple of names $\tilde{a}$ and process Q, whenever $(\nu\tilde{a})(C|Q)$ is stuck-free then also $(\nu\tilde{a})(P|Q)$ is. Our notion of contract refinement differs from stuck-free conformance mainly because we consider a different notion of stuckness. In [FHR$^+$04] a process state is stuck (on a tuple of channel names $\tilde{a}$) if it has no internal moves (but it can execute at least one action on one of the channels in $\tilde{a}$). In our approach, an end-states different from successful termination is stuck (independently of any tuple $\tilde{a}$). Thus, we distinguish between internal deadlock and successful completion while this is not the case in [FHR$^+$04]. Another difference follows from the exploitation of the restriction $(\nu\tilde{a})$; this is used in [FHR$^+$04] to explicitly indicate the local channels of communication used between the contract C and the process Q. In our context we can make a stronger *closed-world* assumption (corresponding to a restriction on all channel names) because service contracts do not describe the entire behaviour of a service, but the flow of execution of its operations inside one session of communication.

The closed-world assumption is considered also in [CCL$^+$06] where, as in our case, a service oriented scenario is considered. In particular, in [CCL$^+$06] a theory of contracts is defined for investigating the compatibility between one client and one service. Our paper consider multi-party composition where several services are composed in a peer-to-peer manner. Moreover, we impose service substitutability as a mandatory property for our notion of refinement; this does not hold in [CCL$^+$06] where it is not in general possible to substitute a service exposing one contract with another one exposing a subcontract. Another relevant difference is that the contracts in [CCL$^+$06] comprises also choices guarded by both input and output actions.

References

[BZ06a] Mario Bravetti and Gianluigi Zavattaro. Contract based Multi-party Service Composition. In *FSEN'07*, volume to appear of LNCS, 2007.

[BGG$^+$05] Nadia Busi, Roberto Gorrieri, Claudio Guidi, Roberto Lucchi, and Gianluigi Zavattaro. Choreography and orchestration: A synergic approach for system design. In *ICSOC'05*, volume 3826 of LNCS, pages 228–240, 2005.

[2] We use $\mathbf{0}$ to denote unsuccessful termination and $\mathbf{1}$ for successful completion.

[BGG$^+$06] Nadia Busi, Roberto Gorrieri, Claudio Guidi, Roberto Lucchi, and Gianluigi Zavattaro. Choreography and orchestration conformance for system design. In *Coordination'06*, volume 4038 of LNCS, pages 63–81, 2006.

[CHY07] Marco Carbone, Kohei Honda, and Nabuko Yoshida. Structured Communication-Centred Programming for Web Services. In *ESOP'07*, volume to appear of LNCS, 2007.

[CCL$^+$06] Samuele Carpineti, Giuseppe Castagna, Cosimo Laneve, and Luca Padovani. A Formal Account of Contracts for Web Services. In *WS-FM'06*, volume 4184 of LNCS, pages 148-162, 2006.

[CL06] Samuele Carpineti and Cosimo Laneve A Basic Contract Language for Web Services. In *ESOP'06*, volume 3924 of LNCS, pages 197–213, 2006.

[DH84] Rocco De Nicola and Matthew Hennessy, Testing Equivalences for Processes. *Theoretical Computer Science*, volume 34: 83–133, 1984.

[FHR$^+$04] Cédric Fournet, C. A. R. Hoare, Sriram K. Rajamani, and Jakob Rehof. Stuck-Free Conformance. In *CAV'04*, volume 3114 of LNCS, pages 242–254, 2004.

[Ley01] F. Leymann. Web Services Flow Language (wsfl 1.0). Technical report, IBM Software Group, 2001.

[RV05] Arend Rensink and Walter Vogler. Fair testing. *CTIT Technical Report TR-CTIT-05-64*, Department of Computer Science, University of Twente, December 2005.

[OAS] OASIS. *Web Services Business Process Execution Language Version 2.0.*

[Tha01] S. Thatte. XLANG: Web services for business process design. Microsoft Corporation, 2001.

[W3C] W3C. *Web Services Choreography Description Language.* `http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217`.