A Theory for Strong Service Compliance[⋆]

Mario Bravetti   Gianluigi Zavattaro

Department of Computer Science, University of Bologna, Italy
E-mail: {bravetti,zavattar}@cs.unibo.it

Abstract. We investigate, in a process algebraic setting, a new notion of compliance that we call *strong service compliance*: composed services are strong compliant if their composition is both deadlock and livelock free (this is the traditional notion of compliance) and whenever a message can be sent to invoke a service, this service is ensured to be ready to serve the invocation. We define also a new notion of refinement, called *strong subcontract pre-order*, suitable for strong compliance: given a composition of strong compliant services each one executing according to some specific contracts, we can replace the services with other services executing corresponding strong subcontracts preserving strong compliance. Finally, we present a characterization of the strong subcontract pre-order resorting to the theory of (should) testing pre-order.

1 Introduction

One of the main novelties emerged during the last years of research in the field of distributed computing is Service Oriented Computing (SOC). It is a novel paradigm 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 contemporarily 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. More precisely, the aim of choreography languages is to support the high level description of systems that should be actually implemented as combination of autonomous, loosely coupled and heterogenous services.

In this paper we continue a formal investigation of service composition initiated in [BZ07]. In particular, in [BZ07] we have presented a process algebraic modeling of the notion of contract intended as a "behavioural interface" of services describing the possible flows of invocations that are received and/or emitted by the service. Based on this contract language, we have formalized the notion of correct composition of services, and we have defined a notion of subcontract preorder characterizing the possibility to replace services with subservices without breaking the correctness of the composition. The notion of correctness considered in [BZ07] requires that the composed services are both deadlock and livelock free.

In this paper we consider a stronger notion, called *strong correctness*, that requires also the following: if a service invocation is ready to be executed, the corresponding invoked service should be ready to serve the request. This intuitive correctness assumption was not taken under consideration is previous work on service compliance such as our previous paper [BZ07] and the paper by Carpineti et al. [CCL+06]. For instance, in these papers, the service $S_1 = \overline{a}_{S_2}|\overline{b}_{S_2}$, invoking in parallel the two operations a and b provided by the service S_2, is considered compliant with a service $S_2 = a;b$ that serves first a and then b. In practice, it could happen that the request on the operation b could reach S_2 before the request on operation a. In this case, usual service invocation protocols (such as, e.g., SOAP) returns an exception indicating the unavailability of the requested operation. *Strong compliance* considers also this kind of exceptions as errors that breaks the correctness of the composition. For instance, according to strong compliance, S_1 is not compliant with the above S_2 but it is compliant with another service $S_2 = a|b$ ready to serve both the requests on a and b.

In order to formalize strong compliance, we need to slightly modify the calculus introduced in [BZ07]. The first difference is that we add locations to services and to output actions. This because the kind of exceptions described above is meaningful assuming that messages are directed to a specific target service, and the exception is raised when the target service is not ready to serve the request contained in the message. The second difference is that the calculus that we consider in the present paper has a standard output prefix instead of the output $\tau;\overline{a}$ always prefixed by an internal τ action considered in [BZ07]. This assumption, permitted us to prove in [BZ07] several interesting results. The most important one is that a suitable subcontract pre-order can be defined abstracting away from the input alphabet of the services in the context. Even if we remove this assumption considering a standard output prefix, the new notion of strong compliance allows us to achieve an even stronger result: a suitable strong subcontract pre-order can be defined abstracting away from both the input and the output alphabet of the other services in the context. This is a rather important property that permits to define a general notion of strong subcontract which is independent of the context in which the corresponding service substitutions are done.

We foresee at least two main applications for our notion of strong subcontract. On the one hand, it can be exploited in the *service discovery* phase. Consider, for instance, a service system defined in terms of the contracts that should be exposed by each of the service components. The actual services to be combined could be retrieved independently one from the other (e.g. querying contemporarily different service registries) collecting those services that either exposes the expected contract, or one of its strong subcontract. On the other hand, the notion of strong subcontract could be useful in *service updates* in order to ensure backward compatibility. Consider, e.g., a service that should be updated in order to provide new functionalities; if the new version exposes a strong subcontract of the previous service, our theory ensures that the new service is a correct substitute for the previous one.

The last technical contribution of the paper is a characterization of our notion of subcontract achieved resorting to the theory of testing [DH84], in particular to the should testing pre-order investigated in [RV05]. This characterization permits to have an effective procedure to prove whether a contract is a strong subcontract of another one. In fact, the definition of strong subcontract is not directly applicable because it contains a universal quantification on all possible contexts.

Structure of the paper. Section 2 reports the syntax and the operational semantics of the calculs that we use in our theory for strong service compliance. Section 3 reports the formalization of strong service compliance while Section 4 reports about our investigation of the new notion of strong subcontract pre-order. Finally, Section 5 contains conclusive remarks and a comparison with the related literature.

2 Contracts and Service Compositions

In this section we introduce the syntax and the operational semantics of the calculus that we use in the following section to investigate formally the notion of strong compliance.

2.1 Syntax

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

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

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

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

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.

The contracts are specified using a typical process algebra. We use $\mathbf{0}$ and $\mathbf{1}$ to denote the two possible final states of a service execution: unsuccessful or successful, respectively. The possible basic actions are the typical internal τ action and the input/output actions. We distinguish between input/output actions executed internally (synchronization between two threads of the same service) and input/output actions involving different services. The first kind of interaction occurs on the contract action names decorated with the subscript star: input actions are denoted with a_*, while output actions with $\overline{a}_*$. The second kind of interaction occurs on the standard actions names: input actions are denoted with a, while output actions with $\overline{a}_l$, where l is the name of the target location of the output action. The composition operators are the standard sequence $\_;\_$, choice $\_+\_$, parallel $\_|\_$, restriction (only on local contract action names) $\_\backslash\_$, and repetition $\_^*$.

The syntax of compositions permits to represent a service located at location l, and executing according to the contract C, simply as $[C]_l$. Services are composed using parallel composition $\_\|\_$ and restriction $\_\backslash\!\backslash\_$. The restriction operator for compositions distinguishes between input and output actions, e.g., we write $[C]_l \backslash\!\backslash \{a_l, \overline{b}_{l'}\}$ to state that the service $[C]_l$ cannot perform inputs on a (e.g., because a is an output port of the service running at l) and cannot perform outputs on the port b of the service running at location l' (e.g., because b is an output port of that service).

2.2 Operational Semantics

The operational semantics is defined in terms of a labeled transition system defined in two steps; we first define the semantics of contracts and, based on the corresponding transition systems, we define the semantics for service compositions.

In the following, 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 1 (plus symmetric rules) while the operational semantics of systems is defined by the rules in Table 2 (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 1. Semantic rules for contracts (symmetric rules omitted).

The operational semantics for contracts is standard for process algebra with sequential composition and repetition. In particular, the label $\surd$ is used to denote the successful completion of a contract. Given $C;D$, the contract D can be activated if C has an outgoing transition labeled with $\surd$; while given the repetition C^*, we have that it can either complete (it has an outgoing transition labeled with $\surd$) or execute one instance of the contract C before becoming C^*

again. The unique specific comment is concerned with the rule for synchronization that is admitted only for input/output actions executed on local contract action names a_*. In fact, the synchronization on global names will be considered in the semantics of systems.

Also the operational semantics for systems is defined in a standard way: the unique nonstandard operator is restriction that, as discussed above, distinguishes between input and output actions executed on the same name. Observe that the synchronization rule considers only nonlocal standard names.

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

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

Table 2. 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., $P \xrightarrow{\varepsilon} P$).

3 Strong Compliance

We now define the notion of strong correct composition of contracts. Intuitively, a composition of services is strongly correct if it is guaranteed that all services eventually reach successful completion (it is both deadlock and livelock free) and everytime a process may invoke an operation on a service, the target service should be ready to serve the request. This second assumption is new with respect to [BZ07] (where we only assumed guaranteed completion) and characterize the new notion of *strong* compliance.

As discussed in the introduction, the rational behind strong compliance is that standard protocols for service invocation usually raise exceptions in the case the target of a service invocation is not ready to serve it. In order to formalize situations in which exception cannot be raised, we define the auxiliary operator $\texttt{nso}(P)$ that evaluates non-synchronizable outputs immediately executable by P, i.e. outputs that do not have a corresponding input, and the predicate $\texttt{exceptionFree}(P)$ that indicates whether none of the above undesired exceptions can be raised in the system P.

Definition 2. (Exception freedomness) *We first define* $\mathtt{nso}(P)$ *inductively on the structure of* P*:*

$$\mathtt{nso}([C]_l) = \{\overline{a}_l \mid C \xrightarrow{\overline{a}_l} C'\}$$
$$\mathtt{nso}(P_1 \| P_2) = (\mathtt{nso}(P_1) - \{\overline{a}_l \mid P_2 \xrightarrow{a_l} P_2'\}) \cup (\mathtt{nso}(P_2) - \{\overline{a}_l \mid P_1 \xrightarrow{a_l} P_1'\})$$
$$\mathtt{nso}(P \backslash\backslash L) = \begin{cases} \{\mathtt{exception}\} & \textit{if } \mathtt{nso}(P) \cap L \neq \emptyset \\ \mathtt{nso}(P) & \textit{otherwise} \end{cases}$$

where `exception` *is an auxiliary name denoting the existence of an output without a corresponding input. Finally, we define:*

$$\mathtt{exceptionFree}(P) \quad \textit{if and only if} \quad \mathtt{nso}(P) = \emptyset$$

Definition 3. (Strongly correct composition) *A system* P *is a strongly correct composition, denoted* $P\Downarrow$*, if for every* P' *such that* $P \xrightarrow{\tau}{}^{*} P'$ *both the following hold:*

- $\mathtt{exceptionFree}(P')$ *and*
- *there exists* P'' *such that* $P' \xrightarrow{\tau}{}^{*} P'' \xrightarrow{\surd}$.

Informally, a system is strongly correct if no exceptions can be raised in any of the reachable states, and the successful termination of all composed services (denoted with the transition labeled with $\surd$) is eventually reached. In [BZ07] we have defined the notion of correct composition, denoted with $P\downarrow$, that corresponds to the above definition without the first item about the exception freedomness of all reachable states.

4 Contract Refinement

In this section we investigate a suitable notion of refinement for contracts compatible with strong correctness; intuitively, a contract C' is a *strong subcontract* of C if it is a "good" substitute of C, i.e. given a system P containing the service $[C]_l$, we can replace it with $[C']_l$ preserving the strong correctness of P.

4.1 Input-Output strong subcontract Relation

In general, see for instance [CCL$^+$06], the subcontract relation depends on the alphabet (i.e. the possible actions) of the services present in P. For instance, we can consider $a + (c; P)$ subcontract of a assuming that the action $\overline{c}_l$ is not in the alphabet of the other services (indeed, this implies that the new branch $c; P$ of the subcontract cannot interfere with the other services in the system).

We start defining a notion of subcontract parameterized on the input and output alphabets of the services in the potential contexts. Then we prove that, thanks to the new exception freedomness assumption, we can abstract away from these alphabets.

We first formally define the input and output alphabets of systems.

Definition 4. (Input and Output sets) *Given the contract $C \in \mathcal{P}_{con}$, we define $I(C)$ as the subset of $\mathcal{N}$ of the potential input actions of C:*

$$\begin{array}{ll} I(\mathbf{0}) = I(\mathbf{1}) = I(\tau) = I(a_*) = I(\overline{a}_*) = I(\overline{a}_l) = \emptyset & I(a) = \{a\} \\ I(C;C') = I(C+C') = I(C|C') = I(C) \cup I(C') & I(C\backslash M) = I(C^*) = I(C) \end{array}$$

We define $O(C)$ as the subset of $\mathcal{N}_{loc}$ of the potential output actions of C:

$$\begin{array}{ll} O(\mathbf{0}) = O(\mathbf{1}) = O(\tau) = O(a) = O(a_*) = O(\overline{a}_*) = \emptyset & O(\overline{a}_l) = \{\overline{a}_l\} \\ O(C;C') = O(C+C') = O(C|C') = O(C) \cup O(C') & O(C\backslash M) = O(C^*) = O(C) \end{array}$$

Note that the set M in $C\backslash M$ does not influence $I(C\backslash M)$ and $O(C\backslash M)$because it contains only contract names outside $\mathcal{N}$. Given the system P, we define $I(P)$ as the subset of $\mathcal{N}_{loc}$ of the potential input actions of P:

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

We define $O(P)$ as the subset of $\mathcal{N}_{loc}$ of the potential output actions of P:

$$O([C]_l) = O(C) \qquad O(P\|P') = O(P) \cup O(P') \qquad O(P\backslash\backslash L) = O(P) - L$$

In the following we make the nonrestrictive assumption that the other services composed in parallel with the service that we want to substitute are of the form $([C_1]_{l_1}\| \cdots \|[C_n]_{l_n})\backslash\backslash L$; this is not restrictive because it is always possible to use standard renaming techniques to avoid capture of names while extending the scope of restrictions. We denote with $\mathcal{P}_{conpres}$ the subset of systems of the form $([C_1]_{l_1}\| \cdots \|[C_n]_{l_n})\backslash\backslash L$.

Note that, given $P = ([C_1]_{l_1}\| \ldots \|[C_n]_{l_n})\backslash\backslash I \cup \overline{O} \in \mathcal{P}_{conpres}$, 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 $O(P) \subseteq O$.

In the next definition we use the following notation: given a contract $C \in \mathcal{P}_{con}$, we use $oloc(C)$ to denote the subset of Loc of the locations target of the output actions occurring inside C.

Definition 5. (Input-Output strong 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 $C' \preceq\!\!\!\preceq_{I,O} C$, if and only if for all $l \in Loc$ such that $l \notin oloc(C) \cup oloc(C')$ and $P \in \mathcal{P}_{conpres,I,O}$ such that $l \notin loc(P)$ we have*

$$([C]_l\|P)\Downarrow \quad \Rightarrow \quad ([C']_l\|P)\Downarrow$$

The following Proposition states an intuitive contravariant property: given $\preceq\!\!\!\preceq_{I',O'}$, and the greater sets I and O (i.e. $I' \subseteq I$ and $O' \subseteq O$) we obtain a smaller relation $\preceq\!\!\!\preceq_{I,O}$ (i.e. $\preceq\!\!\!\preceq_{I,O} \subseteq \preceq\!\!\!\preceq_{I',O'}$). This follows from the fact that extending the sets of input and output actions means considering a greater set of discriminating contexts.

Proposition 1. *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:*

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

Proof. *Let us suppose $C' \preceq_{I,O} C$. Consider now $l \in Loc$, $l \notin oloc(C) \cup oloc(C')$, and $P \in \mathcal{P}_{conpres,I',O'}$, $l \notin loc(P)$, such that $([C]_l \| P)\Downarrow$. As $I' \subseteq I$ and $O' \subseteq O$, then also $P \in \mathcal{P}_{conpres,I,O}$. Thus, as we suppose $C' \preceq_{I,O} C$, $([C']_l \| P)\Downarrow$. This implies that also $C' \preceq_{I',O'} C$.*

The following Proposition states an intermediary result useful in subsequent proofs.

Proposition 2. *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 $l \in Loc$, $l \notin oloc(C) \cup oloc(C')$, and $P \in \mathcal{P}_{conpres,I,O}$, $l \notin loc(P)$, such that $([C]_l \| P)\Downarrow$, we have*

$$\big([C']_l \backslash (I([C']_l) - I([C]_l)) \| P\big)\Downarrow \qquad \text{and} \qquad \big([C']_l \backslash \overline{(O(C') - O(C))} \| P\big)\Downarrow$$

Proof. *We discuss the result concerned with restriction of outputs (the proof for the restriction of inputs is symmetric). 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 $([C']_l \backslash \overline{(O(C') - O(C))} \parallel P)\Downarrow$. We first observe that $([C]_l \parallel P \backslash (O(C') - O(C)))\Downarrow$. Since $C' \preceq_{I,O} C$, we derive $([C']_l \parallel P \backslash (O(C') - O(C)))\Downarrow$.*
As a consequence $([C']_l \backslash \overline{(O(C') - O(C))} \parallel P \backslash (O(C') - O(C)))\Downarrow$. We can conclude $([C']_l \backslash \overline{(O(C') - O(C))} \parallel P)\Downarrow$.

The following proposition states an important property which is a direct consequence of the assumption of exception freedomness of correct compositions.

Proposition 3. *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 $l \in Loc$, $l \notin oloc(C) \cup oloc(C')$, and $P \in \mathcal{P}_{conpres,I,O}$, $l \notin loc(P)$, such that $([C]_l \| P)\Downarrow$,*

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

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 $([C']_l \backslash \overline{(O(C') - O(C))} \parallel P)$, thus reaching the state $([C'_{der}]_l \backslash \overline{(O(C') - O(C))} \parallel P_{der})$. 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, we will have $\texttt{out}([C'_{der}]_l \backslash\backslash \overline{(O(C') - O(C))}) = \{\texttt{exception}\}$, *thus $([C']_l \backslash\backslash \overline{(O(C') - O(C))} \parallel P) \not\Downarrow$ contradicting Proposition 2.*

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, we will have $\texttt{out}(P_{der} \backslash\backslash \overline{(I([C']_l) - I([C]_l))}) = \{\texttt{exception}\}$, *thus $([C']_l \parallel P \backslash\backslash \overline{(I([C']_l) - I([C]_l))}) \not\Downarrow$. This implies $([C']_l \backslash\backslash (I([C']_l) - I([C]_l)) \| P) \not\Downarrow$ contradicting Proposition 2.*

We are finally ready to prove the main results of this subsection. We separate these results in two independent Propositions; the first one states that the set of potential inputs of the other contracts in the system is an information that does not influence the strong subcontract relation, the second one states the same about outputs.

Proposition 4. *Let $C \in \mathcal{P}_{con}$ be a contract, $O \subseteq \mathcal{N}_{loc}$ 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 $C' \in \mathcal{P}_{con}$,*

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

Proof. Let us suppose $C' \preceq_{I',O} C$ (the other direction is symmetric). Given any $l \in Loc$, $l \notin oloc(C) \cup oloc(C')$, 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 3 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 5. *Let $C \in \mathcal{P}_{con}$ be a contract, $O, O' \subseteq \mathcal{N}_{loc}$ 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 $C' \in \mathcal{P}_{con}$,*

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

Proof. Let us suppose $C' \preceq_{I,O'} C$ (the other direction is symmetric). Given any $l \in Loc$, $l \notin oloc(C) \cup oloc(C')$, 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 $([C']_l \backslash\backslash I(C') - I(C) \parallel P \backslash\backslash \overline{(O - I([C]_l))}) \Downarrow$ and $([C']_l \backslash\backslash I(C') -$

$I(C) \parallel P) \Downarrow$. Due to Proposition 3 we have that $([C']_l \backslash\backslash I(C') - I(C) \parallel P)$ 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 $([C']_l \| P) \Downarrow$.

These last two Proposition permits us to forget about the restrictions on the input/output alphabets of the services in the context in which we apply the substitution of one contract with one of its subcontracts, considering always $\preceq_{\mathcal{N}_{loc}, \mathcal{N}_{loc}}$. We denote this relation simply with $\preceq$ and we call it the *strong subcontract pre-order.*[1] We define $\preceq$ assuming a limited set of possible contexts and then we prove that this limitation is not relevant. The new set of contexts does not contain restrictions, i.e., we consider $[C_1]_{l_1} \| \cdots \| [C_n]_{l_n}$ instead of $([C_1]_{l_1} \| \cdots \| [C_n]_{l_n}) \backslash\backslash L$. We call $P \in \mathcal{P}_{conpar}$ the subset of systems of the form $[C_1]_{l_1} \| \cdots \| [C_n]_{l_n}$.

Definition 6. (Strong subcontract pre-order) *A contract C' is a strong subcontract of a contract C denoted $C' \preceq C$, if and only if for all $l \in Loc$ such that $l \notin oloc(C) \cup oloc(C')$ and $P \in \mathcal{P}_{conpar}$ such that $l \notin loc(P)$ we have*

$$([C]_l \| P) \Downarrow \quad \Rightarrow \quad ([C']_l \| P) \Downarrow$$

Proposition 6. *Let $C, C' \in \mathcal{P}_{con}$ be two contracts:*

$$C \preceq C' \qquad \textit{if and only if} \qquad C' \preceq_{\mathcal{N}_{loc}, \mathcal{N}_{loc}} C$$

Proof. The if *part is simple as $\preceq$ is defined as $\preceq_{\mathcal{N}_{loc}, \mathcal{N}_{loc}}$ assuming a subset of possible contexts P.*

We now prove the only-if *part. Supposed $P = ([C_1]_{l_1} \| \ldots \| [C_n]_{l_n}) \backslash\backslash L$, let $I, O \subset \mathcal{N}_{loc}$ be such that $I = \{a_l \mid a_{l_i} \in L \wedge 1 \leq i \leq n\}$ 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 $([C]_l \| (P \backslash\backslash L)) \Downarrow \iff ([C]_l \| (P \backslash\backslash I \cup \overline{O})) \Downarrow \iff ([C]_l \| (P\{\tau; \mathbf{0}/\alpha | \alpha \in \overline{O}\} \backslash\backslash I)) \Downarrow \iff ([C]_l \| P') \Downarrow$, where P' is obtained from $P'' \equiv P\{\tau; \mathbf{0}/\alpha | \alpha \in \overline{O}\}$ 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 strong subcontract pre-order applied to P'.*

4.2 Independent Refinement

In this subsection we prove that the strong subcontract pre-order $\preceq$, which has been defined assuming that the other services in the context are kept unchanged

[1] It is easy to see that $\preceq_{\mathcal{N}_{loc}, \mathcal{N}_{loc}}$ is reflexive and transitive.

while applying the refinement, is suitable also for a more general refinement that is applied independently on all services contemporarily. This is an important property for a refinement notion suitable for service oriented computing; indeed, services are loosely coupled in the sense that they can be updated/modified independently one from the other ones. Independent refinements can be defined as follows.

Definition 7. (Independent subcontract pre-order) *A pre-order $\leq$ over $\mathcal{P}_{con}$ is an independent 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 show that the maximal individual subcontract pre-order corresponds to the pre-order $\preceq\!\!\!\!\!<$ defined in the previous subsection. This is achieved defining a more general class of pre-orders called *singular subcontract pre-orders*, showing that all independent subcontract pre-orders are also *singular subcontract pre-orders*, and finally observing that $\preceq\!\!\!\!\!<$ (which is the maximum of all singular subcontract pre-orders) is also a singular subcontract pre-order.

Intuitively a pre-order $\leq$ over $\mathcal{P}_{con}$ is a singular subcontract pre-order whenever the strong 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 recall that $\mathcal{P}_{conpar}$ denotes the subset of systems of the form $[C_1]_{l_1} \| \ldots \| [C_n]_{l_n}$.

Definition 8. (Singular subcontract pre-order) *A pre-order $\leq$ over $\mathcal{P}_{con}$ is a singular pre-order if, for any $C, C' \in \mathcal{P}_{con}$ such that $C' \leq C$, $l \in Loc$ such that $l \notin oloc(C) \cup oloc(C')$, $P \in \mathcal{P}_{conpar}$ such that $l \notin loc(P)$ we have*

$$([C]_l \| P) \Downarrow \quad \Rightarrow \quad ([C']_l \| P) \Downarrow$$

It is easy to see that the strong subcontract pre-order $\preceq\!\!\!\!\!<$ is the maximal singular subcontract pre-order as it relates all pairs of contracts that satisfy the property stated in the Definition 8.

In order to prove the existence of the maximal independent subcontract pre-order, we will prove that every pre-order that is an independent subcontract is also a singular subcontract (Theorem 1), and vice-versa (Theorem 2).

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

Proof. Suppose that $\leq$ is an independent 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 independent subcontract pre-order, $C_1 = C$, $C'_1 = C'$, $C_2 \dots C_n$ to be such that $P = (C_2 \| \dots \| C_n)$ 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).

Theorem 2. *If a pre-order $\leq$ is a singular subcontract pre-order then it is also an independent subcontract pre-order*

Proof. Consider $n \geq 1$, contracts $C_1, \dots, C_n \in \mathcal{P}_{con}$ and $C'_1, \dots, C'_n \in \mathcal{P}_{con}$ such that $\forall i.\, C'_i \leq C_i$, and distinguished location names $l_1, \dots, l_n \in Loc$ 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 $(P_1 \| \dots \| P_n) \Downarrow$ we can derive $(P'_1 \| \dots \| P'_n) \Downarrow$ in n steps: at the i-th step we replace P_i with P'_i without altering the correctness of the system.

We can, therefore, conclude that there exists a maximal independent subcontract pre-order and it corresponds to "$\preccurlyeq$ ".

4.3 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 strong subcontract relation. This is achieved resorting to the theory of *should-testing* [RV05].

In the following we use the following abuse of notation: "$C \setminus\!\!\setminus M$" stands for "$C\{\mathbf{0}/\alpha | \alpha \in M\}$". This allows us, e.g., to achieve a transition system isomorphic to that of $[C]_l \setminus\!\!\setminus I$, with $I = \{a_l \mid a \in M\}$ for some $M \subseteq \mathcal{N}$, simply by considering $[C \setminus\!\!\setminus M]_l$.

First, we need a preliminary result that is a direct consequence of the fact that $C' \preccurlyeq_{\mathcal{N}_{loc}, \bigcup_{l \in Loc} I([C]_l)} C$ if and only if $C' \preccurlyeq C$, due to Proposition 5.

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

$$C' \setminus\!\!\setminus (I(C') - I(C)) \preccurlyeq C \quad \Rightarrow \quad C' \preccurlyeq C$$

Proof. We will show that the hypothesis yields $C' \preccurlyeq_{\mathcal{N}_{loc}, \bigcup_{l \in Loc} I([C]_l)} C$. From this we can derive the result by using Proposition 5. Given any $l \in Loc$, $l \notin oloc(C) \cup oloc(C')$, 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 $([C' \setminus\!\!\setminus I(C') - I(C)]_l \parallel P) \Downarrow \iff ([C' \setminus\!\!\setminus I(C') - I(C)]_l \parallel P \setminus\!\!\setminus I([C']_l) - I([C]_l)) \Downarrow \iff ([C']_l \parallel P \setminus\!\!\setminus I([C']_l) - I([C]_l)) \Downarrow \iff ([C']_l \| P) \Downarrow$.

Note that the opposite implication trivially holds (by taking $O = \mathcal{N}_{loc}$ and $I = \mathcal{N}_{loc}$ in Proposition 2).

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 $\mathcal{L}$ (i.e. we consider located input and output actions, unlocated input 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 order to resort to the theory defined in [RV05], we first define a transformation on the finite labeled transition system (LTS) of a system P. The trasformation is performed in two steps:

1. First, for every state s of the LTS we do the following: called $I(s)$ the set of labels of outgoing input transitions from the state s, for every label in $I(P) - I(s)$ we add to the state s an outgoing input transition with that label that leads to the $\mathbf{0}$ state.
2. Then, for every output transition $\overline{a}_l$, we replace the output transition with a pair of connected transitions (the first one has the same source as the previous output transition and the second one has the same destination as the previous output transition): a τ transition followed by an $\overline{a}_l$ output transition.

The same transformation is defined also on contracts C. Here, we describe the effect of the two above transformations applied to the contracts checked in the testing scenario that we are going to formalize. The first transformation on the labeled transition system is used to capture those output actions performed by the tester for which there is no corresponding input action in the tested process; the new added input can synchronize with these actions and lead to the state $\mathbf{0}$. This disallows the possibility for the tester to complete its execution, thus to succeed. The second transformation, on the other hand, is necessary to check whether all output actions performable by the tested contract can be received by the tester: in fact, the tau transition added before the output action permits to enter in a state where only the output action is executable. If the tester does not consume this output the contract stucks and the test cannot succeed.

Now we derive a normal form for systems and contracts of our calculus that corresponds to terms of the language in [RV05]. The normal form of the system P (denoted with $\mathcal{NF}(P)$) 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,

$$\mathcal{NF}(P) = rec_{X_1}\theta \qquad \text{where } \theta \text{ is the set of } i\text{-indexed equations}$$

$$X_i = \textstyle\sum_j \lambda_{i,j}; X_{der(i,j)}$$

where, assuming to enumerate the states in the transformed (according to the two-steps procedure above) labeled transition system of P starting from X_1, each variable X_i corresponds to the i-th state of the transformed 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 $X_i = \mathbf{0}$. The normal form of a contract C (denoted with $\mathcal{NF}(C)$) is defined in the same way.

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 $l \in Loc$, $l \notin oloc(C) \cup oloc(C')$, 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 $a_{l'}$ occurring in t into $\overline{a}_{l'}$, and each $\overline{a}_{l'}$ ($\overline{a}$) into $a_{l'}$ (a). From the definition of **shd** *it 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** *and from the fact that output transitions are always preceded by τ transitions (which guarantees that no output can be enabled that does not have a synchronizing input transition) we can conclude that $(C' \backslash\!\!\backslash I(C') - I(C) \| P) \Downarrow$. The thesis directly follows from Lemma 1.*

5 Related Work and Conclusion

We have considered a new notion of correctness for service compositions, modeled using process calculi, in which we assume that whenever a message is sent to a service in order to invoke a particular operation, the service should be ready to serve it. We call this new notion *strong compliance*, and we develop around it an entire theory. It comprises a suitable refinement for services based on a *strong subcontract pre-order*, the proof that this refinement can be applied on each of the service inside a composition independently, and an effective procedure that can be used to prove whether a contract is a subcontract of another one.

The theory of contracts reported in this paper is different from the theory reported in our previous paper [BZ07] in several aspects. The calculus considered in [BZ07] imposes a limitation to output actions that are always preceeded by τ internal actions; on the contrary, in this paper we consider standard input and output prefixes. In this paper we add locations to services and output operations in order to indicate the target of a message; this reflects more faithfully the Web Services technology in which invocations include both the address of the service and the operation to be invoked. Moreover, in this paper we consider a stronger notion of compliance that requires to completely revisit the corresponding notion of (strong) subcontract. The most interesting result is that, even if we consider a more general language than [BZ07], we can achieve even stronger results thanks to the notion of strong compliance. In particular, the strong subcontract pre-order can now be defined abstracting away from both the input and the output alphabets of the services in the context. Moreover, the characterization of the strong subcontract pre-order (achieved also in this paper resorting to the theory of testing) requires new nontrivial technicalities.

It is worth noting that there are some important differences between our form of testing and the traditional one proposed by De Nicola-Hennessy [DH84]. The main difference is that, besides requiring the success of the test, we impose also that the tested process should successfully complete its execution. Moreover, all output actions of both the tester and the tested process should be immediately receivable. Another 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 by 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.

We conclude our analysis of related work considering the theory of contracts proposed 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.

As future work, we plan to investigate strong compliance also in the context of choreography languages, considering a process algebraic modeling of choreographies that follows the proposals by Busi et al. [BGG+05,BGG+06] and by Carbone et al. [CHY07]. In particular, we intend to define a procedure for extracting from a choreography the set of contracts (and strong subcontracts) of services that could correctly play the roles specified in the choreography.

References

[BZ07] 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.

[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.

[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, S. 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*, Dep. of Computer Science, University of Twente, 2005.

[OAS] OASIS. *Web Services Business Process Execution Language Version 2.0.* [http://www.oasis-open.org/committees/download.php/10347/wsbpel-specification-draft-120204.htm].

[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/].