Contract Compliance and Choreography Conformance in the Presence of Message Queues*

Mario Bravetti and Gianluigi Zavattaro

Department of Computer Science, University of Bologna, Italy

Abstract. Choreography conformance and contract compliance have been widely studied in the context of synchronous communication. In this paper we approach a more realistic scenario in which the messages containing the invocations are queued in the called service. More precisely, we study the foundational aspects of contract compliance in a language independent way by just taking contracts to be finite labeled transition systems. Then, we relate the proposed theory of contract compliance with choreography specifications à la WS-CDL where activities are interpreted as pairs of send and receive events. An interesting consequence of adopting a language independent representation of contracts is that choreography projection can be defined in structured operational semantics.

1 Introduction

In the context of Service Oriented Computing (SOC) the problem of the specification of service composition is addressed using two main approaches: service *orchestration* and service *choreography*. According to the first approach, the activities of the composed services are coordinated by a specific component, called the orchestrator, that is responsible for invoking the composed services and collect their responses. Several languages have been already proposed for programming orchestrators such as WS-BPEL [OAS]. As far as choreography languages are concerned, the two main representatives are WS-CDL [W3C] and BPEL4Chor [DKL+07]. Differently from orchestration languages, choreography languages admit the direct interaction among the combined services without the mediation of the orchestrator. In WS-CDL, the basic activity is the interaction between a sender and a receiver, while according to the BPEL4Chor approach a choreography is obtained as the parallel composition of processes that independently execute send and receive activities.

Given an orchestrator (resp. a choreography), one of the main challenges for the SOC community is the definition of appropriate mechanisms for the (semi)automatic retrieval of services that, once combined with the orchestrator (resp. once reciprocally combined), are guaranteed to implement a correct service composition. The currently investigated approach for solving this

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

R. Bruni and K. Wolf (Eds.): WS-FM 2008, LNCS 5387, pp. 37–54, 2009.

problem is to associate to each available service a behavioral description that describes the externally observable message-passing behavior of the service itself. In the literature, this description is known with the name of *behavioral signature* [RR02], *contract* [FHR+04], or (in the specific SOC area) *service contract* [CCL+06, BZ07a, LP07, CGP08]. Assuming that services expose their contract, the above problem can be rephrased as follows: given an orchestrator (resp. a choreography) and a set of service contracts, check whether the services exposing the given contracts can be safely combined with the orchestrator (resp. safely reciprocally combined). The proposed theories of contracts solve this problem formalizing the following notions: *contract compliance* (if a set of contracts is compliant then the corresponding services can be safely combined), *contract refinement* (if a service exposes a refinement of the contract of another service then the former is a safe substitute for the latter), and *choreography conformance* (if the contract of a service is conformant with a given role of a choreography then the service can be used to implement that role in any implementation of the choreography).

In [BZ07b] we have investigated the interplay between the above notions of contract compliance, contract refinement and choreography conformance considering synchronous communication. In this paper we consider a more realistic scenario in which services are endowed with queues used to store the received messages.

More precisely, we revisit our previous theory for contract compliance and choreography conformance [BZ07b] as follows. Contracts are specified in a language independent way by means of finite labeled transition systems. In this way, our new contract theory is more general and foundational as we abstract away from the syntax of contracts and we simply assume that a contract language has an operational semantics defined in terms of a labeled transition system. The presence of queues strongly influences the notion of contract compliance, for instance, the following client and service are now compliant (while this was not the case in [CCL+06, BZ07a, LP07, CGP08]):

$$Client = invoke(a); invoke(b) \qquad Server = receive(b); receive(a)$$

In fact, the presence of queues allows the client to perform the invoke operation in a different order w.r.t. the receive order of the server.

As far as the notion of contract refinement is concerned, the main result is that in the presence of queues refinement can be done independently. That is, given a set of compliant 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 compliant. In general, in a synchronous setting, independent refinement is not possible [CCL+06]. As an example, consider the two following service behaviors:

$$\begin{aligned} Printer &= receive(docToPrint) \\ PrinterFax &= receive(docToPrint) + \\ &\quad receive(docToFax); invoke(faxReceipt) \end{aligned}$$

where + denotes a choice among alternative operations, and the two following client behaviors:

$$\begin{aligned} PrintClient &= invoke(docToPrint) \\ PrintFaxClient &= invoke(docToPrint) + \\ &\quad invoke(docToFax); invoke(faxNum); receive(faxReceipt) \end{aligned}$$

Printer and *PrintClient* can be safely combined. The composition is still correct even if we replace either *Printer* with *PrinterFax* or *PrintClient* with *PrintFaxClient*, but it turns out to be incorrect if we apply both replacements. For this reason we have that in a synchronous context *PrintClient* is not a valid refinement of *Printer*. On the contrary, we will prove that in the presence of message queues the *PrinterFax* service is always a valid refinement of the *Printer* service.

The presence of message queues decouples the send event (corresponding to the introduction of one message in a queue) from the receive event (corresponding to its consumption from the queue). Due to this decoupling, we propose a new interpretation of the semantics of a WS-CDL choreography language in which the two events are modeled by two distinct transitions labeled with a send and a receive label, respectively. Another novelty with respect to previous work is that the choice of representing contracts by means of labeled transition systems allows us to define choreography projections in structured operational semantics. As described below, the use of choreography projection is an important step toward the definition of an appropriate notion of conformance.

Conformance is an important notion to be used to retrieve services that, once combined, correctly implement a given choreography. Formally, (as already done for synchronous communication [BZ07b]) we propose to define conformance as the maximal relation among contracts (ranged over by C), roles (ranged over by r), and choreographies (ranged over by H) written $C \lhd_r H$ such that, given a choreography H with roles $r_1, \cdots, r_n$ and a set of contracts $C_1, \cdots, C_n$ for which $C_1 \lhd_{r_1} H, \cdots, C_n \lhd_{r_n} H$, we have that the composition of $C_1, \cdots, C_n$ is a correct implementation of H. As in our previous work [BZ07b] we show that, unfortunately, there exists no such maximal relation. The proof of this negative result is more complex than in [BZ07b] because, due to the presence of message queues, we had to find out a more subtle counterexample. We partially alleviate this negative result showing that we can define a conformance notion with the above properties as follows: C is conformant to the role r of the choreography H if C is a refinement of the contract obtained by projecting the choreography H to the role r.

Due to space limitations, the proofs of our results are not included in this paper but they can be found in [BZ08].

2 The Theory of Contracts

2.1 Contracts

Contracts are defined as labeled transition systems over located action names, representing operations at a certain location over the network.

Definition 1. *A finite connected labeled transition system (LTS) with termination states is a tuple* $\mathcal{T} = (S, T, L, \longrightarrow, s_0)$ *where* S *is a finite set of states,* $T \subseteq S$ *is a set of states representing successful termination,* L *is a set of labels, the transition relation* $\longrightarrow$ *is a finite subset of* $S \times L \times S$*,* $s_0 \in S$ *and it holds that every state in* S *is reachable (according to* $\longrightarrow$ *from* s_0*).*

Note that non-termination states may have no outgoing transitions: in this case they represent internal failures or deadlocks.

We assume a denumerable set of action names $\mathcal{N}$, ranged over by $a, b, c, \ldots$ and 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. We use $\tau \notin \mathcal{N}$ to denote an internal (unsynchronizable) computation.

Definition 2. *A contract is a finite connected LTS with termination states* $(S, T, L, \longrightarrow, S_0)$*, where* $L = \{a, \overline{a}_l, \tau \mid a \in \mathcal{N}, l \in Loc\}$*, i.e. labels are either a receive (input) on some operation* $a \in N$ *or an invoke (output) directed to some operation* $a \in N$ *at some location* l*.*

In the following we introduce a process algebraic representation for contracts by using a basic process algebra with prefixes over $\{a, \overline{a}_l, \tau \mid a \in \mathcal{N}, l \in Loc\}$ and we show that from the LTS denoting a contract we can derive a process algebraic term whose behavior is the same as that of the LTS. The process algebra is a simple extension of basic CCS [Mil89] with successful termination denoted by "**1**": this new term is necessary in order to have two kinds of states without outgoing transitions, those that are successfully terminating (that we denote with the process "**1**") and those that are not (denoted with the traditional null process "**0**").

Definition 3 (Contracts). *We consider a denumerable set of contract variables* Var *ranged over by* $X, Y, \cdots$*. The syntax of contracts is defined by the following grammar*

$$\begin{array}{lcl} C & ::= & \mathbf{0} \mid \mathbf{1} \mid \alpha.C \mid C+C \mid X \mid recX.C \\ \alpha & ::= & \tau \mid a \mid \overline{a}_l \end{array}$$

where $recX.\_$ *is a binder for the process variable* X *denoting recursive definition of processes. The set of the contracts* C *in which all process variables are bound, i.e.* C *is a closed term, is denoted by* $\mathcal{P}_{con}$*.*

Besides the already commented recursion operator, we consider the standard prefix $\alpha.\_$ (with possible prefixes τ, a, and $\overline{a}_l$ denoting internal, input, and output action, respectively) and choice $\_ + \_$ operators. In the following we will omit trailing "**1**" when writing contracts.

The structured operational semantics of contracts is defined in terms of a transition system labeled over $L = \{a, \overline{a}_l, \tau, \mid a \in \mathcal{N}, l \in Loc\}$ and a termination predicate $\surd$ over states obtained by the rules in Table 1 (plus symmetric rule for choice).

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

$$\mathbf{1}\surd \qquad\qquad \alpha.C \xrightarrow{\alpha} C$$

$$\frac{C \xrightarrow{\alpha} C'}{C+D \xrightarrow{\alpha} C'} \qquad\qquad \frac{C\surd}{C+D\surd}$$

$$\frac{C\{recX.C/X\} \xrightarrow{\alpha} C'}{recX.C \xrightarrow{\alpha} C'} \qquad\qquad \frac{C\{recX.C/X\}\surd}{recX.C\surd}$$

Note that we use the notation $C\{recX.C/X\}$ to denote syntactic replacement of free occurrences of variable X in C with the same contract C (where, as usual, α-conversion is applied to avoid the possible captures of variable names). The rules for the operational semantics are standard; we simply comment the actual meaning of the termination predicate $\surd$. Informally, a contract C satisfies the predicate if it is the successfully terminating terms $\mathbf{1}$ or it is a more complex term in which there is at least one $\mathbf{1}$ that does not occur inside a prefixed term $\alpha.C$.

We have that the semantics of a contract $C \in \mathcal{P}_{con}$ gives rise to a finite connected LTS with termination states $(S, T, L, \longrightarrow, C)$ where $L = \{a, \overline{a}_l, \tau, \mid a \in \mathcal{N}, l \in Loc\}$ and: S is the set of states reachable from C, T is the subset of S of the states for which the predicate $\surd$ is true and $\longrightarrow$ includes only transitions between states of S. Note that the fact that such a LTS is finite (i.e. finite-state and finitely branching) is a well-known fact for basic CCS [Mil89] (and obviously the additional presence of successful termination does not change this fact).

Definition 4. *A set of process algebraic equations is denoted by $\theta = \{X_i = C_i \mid 0 \leq i \leq n-1\}$, where n is the number of equation in the set, X_i are process variables, and C_i are contract terms (possibly including free process variables). The process algebraic equations θ is closed if only process variables X_i, with $0 \leq i \leq n-1$, occur free in the bodies C_j, with $0 \leq j \leq n-1$, of the equations in the set.*

Definition 5. *Let $\mathcal{T} = (S, T, L, \longrightarrow, S_0)$ be a contract. A contract term $C \in \mathcal{P}_{con}$ is obtained from $\mathcal{T}$ as follows.*

- *Supposed $S = \{s_0, \ldots, s_{n-1}\}$ (i.e. any given numbering on the states S), we first obtain from $\mathcal{T}$ a finite closed set of equations $\theta = \{X_i = C_i \mid 0 \leq i \leq n-1\}$ as follows. Denoted by m_i the number of transitions outgoing from s_i, by α_j^i the label of the $j-th$ transition outgoing from s_i (for any given numbering on the transitions outgoing from s_i), with $j \leq m_i$, and by $s_{succ_j^i}$ its target state, we take $C_i = \sum_{j \leq m_i} \alpha_j^i.X_{succ_j^i} + \{\mathbf{1}\}$, where $\mathbf{1}$ is present only if $s_i \in T$ and an empty sum is assumed to yield $\mathbf{0}$.*
- *We then obtain, from the closed set of equations $\theta = \{X_i = C_i \mid 0 \leq i \leq n-1\}$, a closed contract term C by induction on the number of equations. The base case is $n = 1$: in this case we have that C is $recX_0.C_0$. In the inductive*

case we have that C is inductively defined as the term obtained from the equation set $\{X_i = C'_i \mid 0 \leq i \leq n-2\}$, where $C'_i = C_i\{recX_{n-1}.C_{n-1}/X_{n-1}\}$.

Definition 6. *A homomorphism from a finite connected LTS with termination states $\mathcal{T} = (S, T, L, \longrightarrow, s_0)$ to a finite connected LTS with termination states $\mathcal{T}' = (S', T', L, \longrightarrow', s'_0)$ is a function f from S to S' such that: $f(s_0) = s'_0$ and for all $s \in S$ we have $\{(l, s') \mid f(s) \stackrel{l}{\longrightarrow}' s'\} = \{(l, f(s')) \mid s \stackrel{l}{\longrightarrow} s'\}$, i.e. the set of transitions performable by $f(s)$ is the same as the set of transitions performable by s when f-images of the target states are considered, and $s \in T$ if and only if $f(s) \in T'$.*

Note that, if f is a homomorphism between finite connected LTSes with finite states then f is surjective: this because all states reachable by $f(s_0)$ must be f-images of states reachable from s_0.

Proposition 1. *Let $\mathcal{T} = (S, T, L, \longrightarrow, s_0)$ be a contract and $C \in \mathcal{P}_{con}$ be a contract term obtained from $\mathcal{T}$. There exists a (surjective) homomorphism from the semantics of C to $\mathcal{T}$ itself.*

2.2 Composing Contracts

Definition 7 (Systems). *The syntax of systems (compositions of contracts) is defined by the following grammar*

$$\begin{array}{lcl} P & ::= & [C, \mathcal{Q}]_l \quad \mid \quad P \| P \quad \mid \quad P \backslash L \\ \mathcal{Q} & ::= & \epsilon \quad \mid \quad a^l :: \mathcal{Q} \end{array}$$

where $L \subseteq \mathcal{N}_{loc}$.

The restriction operator $\_\backslash L$ is a binder for the names in located actions. Formally, if a_l is in L, then L binds a in any action a occurring in the contract located at l and in any action $\overline{a}_l$. The terms in the syntactic category $\mathcal{Q}$ denote message queues. They are lists of messages, each one denoted with a^l where a is the action name and l is the location of the sender. We use ϵ to denote the empty message queue. Trailing ϵ are usually left implicit, and we use :: also as an operator over the syntax: if $\mathcal{Q}$ and $\mathcal{Q}'$ are ϵ-terminated queues, according to the syntax above, then $\mathcal{Q} :: \mathcal{Q}'$ means appending the two queues into a single ϵ-terminated list. Therefore, if $\mathcal{Q}$ is a queue, then $\epsilon :: \mathcal{Q}$, $\mathcal{Q} :: \epsilon$, and $\mathcal{Q}$ are syntactically equal.

A system P is well-formed if: (i) every contract subterm $[C, \mathcal{Q}]_l$ 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 $[C, \mathcal{Q}]_l$ 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. Moreover, we will use the shorthand $[C]_l$ to stand for $[C, \epsilon]_l$.

Given a system P, we use $loc(P)$ to denote the subset of Loc of the locations of contracts syntactically occurring inside P: e.g. $loc([C]_{l_1} \| [C']_{l_2}) = \{l_1, l_2\}$.

Also the operational semantics of systems is defined in terms of a labeled transition system. The labels, denoted with λ, λ', $\cdots$, are taken from the set $\{a_{rs}, \overline{a}_{rs}, a^{+}_{r\to s}, a^{-}_{r\to s}, \tau \mid a \in \mathcal{N}; r, s \in Loc\}$, where: a_{rs} denotes a potential input by a queue where the sender is at location r and the receiver queue is at location s, $\overline{a}_{rs}$ denotes a potential output where the sender is at location r and the receiver queue is at location s, $a^{+}_{r\to s}$ denotes an insertion in the queue (that actually took place) where the sender is at location r and the receiver queue is at location s, $a^{-}_{r\to s}$ denotes an extraction from the queue (that actually took place) where the sender (that originally sent the message) is at location r and the receiver queue is at location s, and τ denotes a move performed internally by one contract in the system. We use $\alpha-$renaming of names bound by the restriction operator $\_\backslash L$; namely, we write $P \equiv_a Q$ if P is α-convertible into Q (or vice-versa), i.e. if Q can be obtained from P by turning subterms $P'\backslash L$ of P into subterms $Q'\backslash L'$ by renaming of located names a_l of L into located names $ren(a)_l$ (yielding L' with the same cardinality) and by correspondingly replacing: (i) each input-related syntactical occurrence of a with $ren(a)$ inside the unique subterm $[C, \mathcal{Q}]_l$ of P', if it exists (more precisely occurrences of $a^{l'}$ inside $\mathcal{Q}$ are renamed into $ren(a)^{l'}$, independently of the location l', and a input prefixes inside C are renamed into $\underline{ren(a)}$ input prefixes), and (ii) each syntactical occurrence of $\overline{a}_l$ inside P' with $\overline{ren(a)}_l$ (obviously a renaming is only allowed if it does not generate a name that is already present as a free name in association with the same location).

The rules in the Table 2 (plus symmetric rules) define the transition system and the termination predicate ($\surd$) for systems. In Table 2 we assume that $a^l \in \mathcal{Q}$ holds true if and only if a^l syntactically occurs inside $\mathcal{Q}$.

Table 2. Semantic rules for contract compositions (symmetric rules omitted)

$$[C, \mathcal{Q}]_s \xrightarrow{a_{rs}} [C, \mathcal{Q} :: a^r]_s \qquad \frac{C \xrightarrow{\overline{a}_s} C'}{[C, \mathcal{Q}]_r \xrightarrow{\overline{a}_{rs}} [C', \mathcal{Q}]_r} \qquad \frac{P \xrightarrow{a_{rs}} P' \quad Q \xrightarrow{\overline{a}_{rs}} Q'}{P \| Q \xrightarrow{a^{+}_{r\to s}} P' \| Q'}$$

$$\frac{C \xrightarrow{\tau} C'}{[C, \mathcal{Q}]_l \xrightarrow{\tau} [C', \mathcal{Q}]_l} \qquad \frac{C\surd}{[C, \epsilon]_l\surd} \qquad \frac{P \xrightarrow{\lambda} P'}{P \| Q \xrightarrow{\lambda} P' \| Q} \qquad \frac{P\surd \quad Q\surd}{P \| Q\surd}$$

$$\frac{P \xrightarrow{\lambda} P' \quad \text{if } \lambda = a_{rs}, \overline{a}_{rs} \text{ then } a_s \notin L}{P\backslash L \xrightarrow{\lambda} P'\backslash L} \qquad \frac{P\surd}{P\backslash L\surd}$$

$$\frac{P \equiv_\alpha P' \quad P' \xrightarrow{\lambda} Q}{P \xrightarrow{\lambda} Q} \qquad \frac{C \xrightarrow{a} C' \quad \text{if } b^l \in \mathcal{Q} \text{ then } b \neq a}{[C, \mathcal{Q} :: a^r :: \mathcal{Q}']_s \xrightarrow{a^{-}_{r\to s}} [C', \mathcal{Q} :: \mathcal{Q}']_s}$$

We will also 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., $P \xrightarrow{\varepsilon} P$). In the following we will adopt the usual notation A^* to denote (possibly empty) sequences over labels in A.

We now define the notion of correct composition of contracts. This notion is the same as in [BZ07a]. 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 8 (Correct contract composition). *A system P is a correct contract composition, denoted $P\downarrow$, if for every P' such that $P \xrightarrow{w} P'$, with $w \in \{a^+_{r\to s}, a^-_{r\to s}, \tau \mid a \in \mathcal{N}; r, s \in Loc\}^*$, there exists P'' such that $P' \xrightarrow{w'} P''$, with $w' \in \{a^+_{r\to s}, a^-_{r\to s}, \tau \mid a \in \mathcal{N}; r, s \in Loc\}^*$, and $P''\surd$.*

It is interesting to observe that in a correct contract composition, when all contracts successfully terminate, it is ensured that all the sent messages have been actually received. In fact, by definition of the termination predicate $\surd$ for contract compositions, a system is terminated only if all message queues are empty. Note also that, obviously, contracts that form correct contract compositions still form correct contract compositions if they are replaced by homomorphic ones.

We complete this subsection presenting a simple example of correct contract composition

$$[\overline{a}_{l_3}]_{l_1} \parallel [\overline{b}_{l_3}]_{l_2} \parallel [a.b]_{l_3}$$

composed by three contracts, the first one and the second one that send respectively the message a and b to the third one, and this last contract that consumes the two messages.

2.3 Independent Subcontracts

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.

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 $P \stackrel{w}{\Longrightarrow} P'$ if there exist $P_1, \cdots, P_m$ such that $P \xrightarrow{\tau^*} P_1 \xrightarrow{\lambda_1} P_2 \xrightarrow{\tau^*} \cdots \xrightarrow{\tau^*} P_{m-1} \xrightarrow{\lambda_n} P_m \xrightarrow{\tau^*} P'$.

Definition 9 (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 $\forall i.\, oloc(C_i) \cup oloc(C'_i) \subseteq \{l_j \mid 1 \leq j \leq n \wedge j \neq i\}$, we have $([C_1]_{l_1} \parallel \ldots \parallel [C_n]_{l_n})\downarrow$ implies*

– $([C'_1]_{l_1} \| \ldots \| [C'_n]_{l_n})\downarrow$ *and*
– $\forall\, w \in \{a^+_{r\to s}, a^-_{r\to s} \mid a \in \mathcal{N}; r, s \in Loc\}^*.$
$\exists P' : ([C'_1]_{l_1} \| \ldots \| [C'_n]_{l_n}) \stackrel{w}{\Longrightarrow} P' \wedge P'\surd \quad \Rightarrow$
$\exists P'' : ([C_1]_{l_1} \| \ldots \| [C_n]_{l_n}) \stackrel{w}{\Longrightarrow} P'' \wedge P''\surd.$

We will prove that there exists a maximal independent subcontract pre-order; this is a direct consequence of the queue based communication. In fact, if we simply consider synchronous communication it is easy to prove that there exists no maximal independent subcontract pre-order (see [BZ07a]).

We will show that the maximal independent subcontract pre-order 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 that $([C_1]_{l_1} \| \ldots \| [C_i]_{l_i} \| \ldots \| [C_n]_{l_n})\downarrow$ implies that the statement in Def. 9 holds for $([C_1]_{l_1} \| \ldots \| [C'_i]_{l_i} \| \ldots \| [C_n]_{l_n})$. 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 10 (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 $C' \leq C$, $l \in Loc$ and $P \in \mathcal{P}_{conpar}$ such that $l \notin loc(P)$ and $oloc(C) \cup oloc(C') \subseteq loc(P)$, we have $([C]_l \| P)\downarrow$ implies*

– $([C']_l \| P)\downarrow$ *and*
– $\forall\, w \in \{a^+_{r\to s}, a^-_{r\to s} \mid a \in \mathcal{N}; r, s \in Loc\}^*.$
$\exists P' : ([C']_l \| P) \stackrel{w}{\Longrightarrow} P' \wedge P'\surd \quad \Rightarrow \quad \exists P'' : ([C]_l \| P) \stackrel{w}{\Longrightarrow} P'' \wedge P''\surd.$

The following proposition, which shows that extending possible contexts with an external restriction does not change the notion of singular subcontract pre-order, will be used in the following Sect. 2.4. It plays a fundamental role in eliminating the source of infinite branching in the interaction behavior of the contract composition originated by α-renaming of restriction. We let $\mathcal{P}_{conpres}$ denote the set of systems of the form $([C_1]_{l_1} \| \ldots \| [C_n]_{l_n}) \backslash L$, with $C_i \in \mathcal{P}_{con}$ for all $i \in \{1, \ldots, n\}$ and $L \subseteq \mathcal{N}_{loc}$.

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

– $([C']_l \| P)\downarrow$ *and*
– $\forall\, w \in \{a^+_{r\to s}, a^-_{r\to s} \mid a \in \mathcal{N}; r, s \in Loc\}^*.$
$\exists P' : ([C']_l \| P) \stackrel{w}{\Longrightarrow} P' \wedge P'\surd \quad \Rightarrow \quad \exists P'' : ([C]_l \| P) \stackrel{w}{\Longrightarrow} P'' \wedge P''\surd.$

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

Definition 11 (Subcontract relation). *A contract C' is a subcontract of a contract C denoted $C' \preceq C$, if and only if for all $l \in Loc$ and $P \in \mathcal{P}_{conpar}$ such that $l \notin loc(P)$ and $oloc(C) \cup oloc(C') \subseteq loc(P)$, we have that $([C]_l \| P)\downarrow$ implies*

– $([C']_l \| P)\downarrow$ *and*
– $\forall\, w \in \{a^+_{r\to s}, a^-_{r\to s} \mid a \in \mathcal{N}; r, s \in Loc\}^*.$
$\exists P' : ([C']_l \| P) \stackrel{w}{\Longrightarrow} P' \wedge P'\surd \quad \Rightarrow \quad \exists P'' : ([C]_l \| P) \stackrel{w}{\Longrightarrow} P'' \wedge P''\surd.$

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.

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.*

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

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

2.4 Input-Output Knowledge Independence

In the following we will show that allowing the subcontract relation to depend on the knowledge about input and output actions of other initial contracts does not change the relation. As a consequence of this fact we will show that input on new types (operations) can be freely added in refined contracts.

Given a set of located action names $I \subseteq \mathcal{N}_{loc}$, 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 12 (Input and Output sets). *Given a 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):*

$$I(\mathbf{0}) = I(\mathbf{1}) = I(X) = \emptyset \qquad I(a.C) = \{a\} \cup I(C)$$
$$I(C+C') = I(C) \cup I(C') \qquad I(\overline{a}_l.C) = I(\tau.C) = (recX.C) = I(C)$$

Given a system $P \in \mathcal{P}_{conpres}$, 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):

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

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

- $([C']_l \| P)\downarrow$ *and*
- $\forall\, w \in \{a^+_{r \to s}, a^-_{r \to s} \mid a \in \mathcal{N}; r, s \in Loc\}^*$.
  $\exists P' : ([C']_l \| P) \stackrel{w}{\Longrightarrow} P' \wedge P'\surd \quad \Rightarrow \quad \exists P'' : ([C]_l \| P) \stackrel{w}{\Longrightarrow} P'' \wedge P''\surd$.

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 located names such that $I' \subseteq I$ and $O, O' \subseteq \mathcal{N}_{loc}$ be two sets of output located names such that $O' \subseteq O$. We have:*

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

The following lemma, that will be used to characterize the subcontract relation, states that a subcontract is still a subcontract even if we modify it so to consider only the inputs and outputs already available in the supercontract.

In the following lemma, and in the remainder of the paper, we use the abuse of notation "$C \backslash M$", with $M \subseteq \mathcal{N}$, to stand for "$C\{\mathbf{0}/\alpha.C' | \alpha \in M\}$", that denotes the effect of restricting C with respect to inputs in M.

Lemma 1. *Let $C, C' \in \mathcal{P}_{con}$ be contracts and $I, O \subseteq \mathcal{N}_{loc}$ be sets of located names. We have that both the following hold*

$$\begin{array}{lll} C' \preceq_{I,O} C & \Rightarrow & C' \backslash (I(C') - I(C)) \preceq_{I,O} C \\ C' \preceq_{I,O} C & \Rightarrow & C'\{\tau.\mathbf{0}/\alpha.C'' \mid \alpha \in \overline{O(C') - O(C)}\} \preceq_{I,O} C \end{array}$$

A fundamental result depending on the queue based communication 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 (and similarly for the system considered as context).

Proposition 4. *Let $C, C' \in \mathcal{P}_{con}$ be contracts and $I, O \subseteq \mathcal{N}_{loc}$ be sets of located names. Let $l \in Loc$ and $P \in \mathcal{P}_{conpres,I,O}$, $l \notin loc(P)$ and $oloc(C) \cup oloc(C') \subseteq loc(P)$ be such that $([C]_l \| P)\downarrow$. We have that both the following hold:*

If $([C'\{\tau.\mathbf{0}/\alpha.C'' \mid \alpha \in \overline{O(C') - O(C)}\}]_l \| P)\downarrow$ *then*

$$([C']_l \| P) \xrightarrow{w} ([C'_{der}, \mathcal{Q}]_l \| P_{der}) \wedge w \in \{a^+_{r \to s}, a^-_{r \to s}, \tau \mid a \in \mathcal{N}; r, s \in Loc\}^* \Rightarrow \forall\, a_{l'} \in O(C') - O(C).\, C'_{der} \xrightarrow{\overline{a}_{l'}}\!\!\!\!\!/$$

If $([C'\backslash(I(C') - I(C))]_l \| P)\downarrow$ *then*

$$([C']_l \| P) \xrightarrow{w} ([C'_{der}, \mathcal{Q}]_l \| P_{der}) \wedge w \in \{a^+_{r \to s}, a^-_{r \to s}, \tau \mid a \in \mathcal{N}; r, s \in Loc\}^* \Rightarrow \forall\, a \in I(C') - I(C). \forall\, r \in loc(P).\, P_{der} \xrightarrow{\overline{a}_{rl}}\!\!\!\!\!/$$

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 5. *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 $C' \in \mathcal{P}_{con}$,*

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

Proposition 6. *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 $C' \in \mathcal{P}_{con}$,*

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

We finally show that the subcontract relation $\preceq$ allows input on new types (and unreachable outputs on new types) to be added in refined contracts. The result, that uses Lemma 1, is a direct consequence (in the case of inputs) of the fact that $C' \preceq_{\mathcal{N}_{loc}, \bigcup_{l \in Loc} I([C]_l)} C$ if and only if $C' \preceq C$, i.e. it exploits the results above about independence from knowledge of types used by other initial contracts.

Theorem 3. *Let $C, C' \in \mathcal{P}_{con}$ be contracts. Both the following hold*

$$C'\backslash(I(C') - I(C)) \preceq C \quad \iff \quad C' \preceq C$$
$$C'\{\tau.\mathbf{0}/\alpha.C'' \mid \alpha \in \overline{O(C') - O(C)}\} \preceq C \quad \iff \quad C' \preceq C$$

3 Contract-Based Choreography Conformance

We first introduce a choreography language similar to those already presented in [BGG+05, CHY07, BZ07b]. The main novelty is that, as we are considering communication mediated by a message queue, in the operational semantics we distinguish between the send and the receive events.

Definition 14 (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 $a_{r\to s}$ *(where we assume* $r \neq s$*) 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 three auxiliary terms $a^-_{r\to s}$, $\mathbf{1}$, and $\mathbf{0}$. The first one is used to model the fact that an asynchronous interaction has been activated but not yet completed. The other two terms 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 3 where we take η to range over the set of labels $\{a^+_{r\to s}, a^-_{r\to s} \mid a \in Operations, r, s \in Roles\}$ and the termination predicate $\surd$. The rules in Table 3 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.

Table 3. Semantic rules for choreographies (symmetric rules omitted)

$$a_{r\to s} \xrightarrow{a^+_{r\to s}} a^-_{r\to s} \qquad a^-_{r\to s} \xrightarrow{a^-_{r\to s}} \mathbf{1} \qquad \mathbf{1}\surd \qquad H^*\surd$$

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

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

Choreographies are especially useful to describe the protocols of interactions within a group of collaborating services, nevertheless, 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 define a mechanism to extract from a choreography the description of the behavior of a given role. Formally, for each role i, we define a labeled transition system with transitions $\xrightarrow{\eta}_i$ (see the

rules in Table 4) and termination predicate $\surd_i$ representing the behavior of the role i. In the following, given a choreography H and one of its role i, with $semH_i$ we denote the contract term obtained from the labeled transition system $transn_i$ according to the technique defined in Section 2.

Table 4. Projection on the role i of a choreography (symmetric rules omitted)

$$a_{r\to s} \xrightarrow{\overline{a}_s}_r \mathbf{1} \qquad a_{r\to s} \xrightarrow{a}_s \mathbf{1} \qquad a_{r\to s}\surd_i \text{ if } i \neq r,s$$

$$\mathbf{1}\surd_i \qquad H^*\surd_i$$

$$\frac{H \xrightarrow{\eta}_i H'}{H+L \xrightarrow{\eta}_i H'} \qquad \frac{H\surd_i}{H+L\surd_i} \qquad \frac{H \xrightarrow{\eta}_i H'}{H;L \xrightarrow{\eta}_i H';L} \qquad \frac{H\surd_i \quad L \xrightarrow{\eta}_i L'}{H;L \xrightarrow{\eta}_i L'}$$

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

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., $Roles = Loc$. Taken these assumptions, we have that the labels of the operational semantics of the choreography calculus are a subset of the labels of the operational semantics of contract systems, i.e. $a^+_{r\to s}$ and $a^-_{r\to s}$.

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 15 (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 $a^+_{r\to s}$ and $a^-_{r\to s}$, if $P \stackrel{w}{\Longrightarrow} P'$ and $P'\surd$ then there exists H' such that $H \stackrel{w}{\longrightarrow} H'$ and $H'\surd$.*

Note that it is not necessary for an implementation to include all possible conversations admitted by a choreography. As an example, consider the choreography $reserve_{client\to server};(accept_{server\to client} + reject_{server\to client})$. We can think of implementing it with the following system

$$[\overline{reserve}_{server}.(accept + reject)]_{client} \parallel [reserve.\overline{accept}_{client}]_{server}$$

where the server is always ready to accept the client's request.

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 $a_{r\to s}\ ;\ b_{t\to u}$. 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 $b^+_{t\to u} b^-_{t\to u} a^+_{r\to s} a^-_{r\to s}$ 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. We restrict our interest to well formed choreographies.

Definition 16 (Well formed choreography). *A choreography H, defined on the roles $r_1, \cdots, r_n$, is* well formed *if* $[\![H]\!]_{r_1}]_{r_1} \parallel \cdots \parallel [\![H]\!]_{r_n}]_{r_n} \propto H$

As another example of non well formed choreography we consider $a_{l_1\to l_3}; b_{l_2\to l_3}$ which have the following projection $[\overline{a}_{l_3}]_{l_1} \parallel [\overline{b}_{l_3}]_{l_2} \parallel [a.b]_{l_3}$ corresponding to the system described at the end of the subsection 2.2. Among the possible traces of this system we have $a^+_{l_3} b^+_{l_3} a^-_{l_3} b^-_{l_3}$ which is not a correct trace for the above choreography. This example is of interest because it shows that some interesting contract systems are not specifiable as choreographies. This follows from the fact that we have adopted the same approach of WS-CDL that exploits synchronizations as its basic activity. In order to model at a choreographic level the above contract system, we should separate also in the syntax (and not only in the semantics) the send from the receive actions. For instance, we could consider two distinct basic terms $a^+_{r\to s}$ and $a^-_{r\to s}$ for send and receive actions, respectively, and describe the above system with the choreography $a^+_{l_1\to l_3} \mid b^+_{l_2\to l_3} \mid a^-_{l_1\to l_3}; b^-_{l_2\to l_3}$.

We are now in place for the definition of the relation $C \lhd_r H$ indicating whether the contract C can play the role r in the choreography H.

Definition 17 (Conformance family). *A relation among contracts, roles, and choreographies denoted with $C \lhd_r H$ is a* conformance relation *if, given a well formed choreography H with roles $r_1, \cdots, r_n$, we have that $[\![H]\!]_{r_i} \lhd_{r_i} H$ for $1 \leq i \leq n$ and if $C_1 \lhd_{r_1} H, \cdots, C_n \lhd_{r_n} H$ then $[C_1]_{r_1} \| \cdots \| [C_n]_{r_n} \propto H$*

In the case of synchronous communication we proved in [BZ07a] a negative result about conformance: differently from the subcontract pre-orders defined on contracts in the previous Section, there exists no maximal conformance relation. The counter-example used in that paper to prove this negative results does not work in the presence of message queues, but we have found out the following more subtle counter-example. Consider the choreography $H = a_{r\to s} | b_{s\to r}$. We could have two different conformance relations, the first one $\lhd^1$ including (besides the projections) also $a.\overline{b}_r \lhd^1_s H$ and the second one $\lhd^2$ including also $b.\overline{a}_s \lhd^2_r H$. It is easy to see that it is not possible to have a conformance relation that comprises the union of the two relations $\lhd^1$ and $\lhd^2$. In fact, the system $[b.\overline{a}_s]_r \parallel [a.\overline{b}_r]_s$ is not a correct composition because the two contracts are both blocked for a never incoming message.

The remainder of the paper is dedicated to the definition of a mechanism that, exploiting the choreography projection and the notion of contract refinement

defined in the previous Section, permits to characterize an interesting conformance relation. This relation is called *consonance*.

Definition 18 (Consonance). *We say that the contract C is* consonant *with the role r of the well formed choreography H (written $C \bowtie_r H$) if $C \preceq [\![H]\!]_r$ where $\preceq$ is the subcontract relation defined in Section 2.*

Theorem 4. *Given a well formed choreography H, we have that the consonance relation $C \bowtie_r H$ is a conformance relation.*

4 Related Work and Conclusion

We have addressed the problem of the definition of suitable notions of contract refinement and choreography conformance for services that communicate through message queues. We have attacked this problem exploiting the approach that we have already successfully adopted for synchronously communicating services [BZ07b]. However, the new theory of contracts is more general than the theory in our previous paper because we represent contracts in a language independent way. On the one hand, this required to significantly revisit our technical contribution, but on the other hand, our results are now more general as they apply to any contract language (for which an operational semantics is defined in terms of a labeled transition system). This choice also influenced the theory for choreography conformance. Now a choreography projection must produce a labeled transition system instead of a contract specified in a given language. We solve this problem defining the projection in structured operational semantics.

It is worth noting that, differently from our previous work, in this paper we do not present an actual way for deciding compliance, refinement, and conformance. This follows from the fact that the presence of message queues make a contract system possibly infinite. In fact, even if contracts are finite state, a contract could repeatedly emit the same message thus introducing an unbounded amount of messages in a queue. Contract systems can be limited to be finite in (at least) two possible ways, either considering bounded buffers or avoiding cycles in contracts.

In the Introduction we have already commented similar contract theories available in the literature [CCL$^+$06, LP07, CGP08] developed for synchronous communication. Similar ideas were already considered also in [FHR$^+$04] where the notion of *stuck-free* conformance is introduced. The unique contract theories for asynchronous communication that we are aware of are by Rajamani and Rehof [RR02] and by van der Aalst and others [ALM$^+$07]. In [RR02] a conformance relation is defined in a bisimulation-like style introducing an ad-hoc treatment of internal and external choices that are included in the calculus as two distinct operators. We try somehow to be more general, avoiding the introduction of two distinct choice operators and by defining our refinement notion indirectly as the maximal contract substitution relation that preserves system correctness. In [ALM$^+$07] the same approach for formalizing compliance and refinement that we have presented in [BZ07b] has been applied to service systems

specified using open Workflow Nets (a special class of Petri nets) that communicate asynchronously. As in our works, they prove that contract refinement can be done independently. Moreover, they present an actual way for checking refinement that work assuming that contracts do not contain cycles. As a future work, we plan to investigate whether their decidability technique can be applied also in our different context in which message queues preserve the order of messages.

We now comment on the testing theories developed for process calculi starting from the seminal work by De Nicola and Hennessy [DH84]. A careful comparison between the testing approach and our contract theory for synchronous communication can be found in [BZ07a] (where we resort to fair testing [RV07], a variant of De Nicola-Hennessy must testing for fair systems, to define an actual procedure to check contract refinement). The same comments apply also to the CSP failure refinement [Hoa85] as it is well known that the must testing pre-order and the CSP failure refinement coincide (at least for finitely branching processes without divergences) [DeN87]. As far as must testing for asynchronous communication is concerned, it has been investigated for asynchronous CCS in [CH98, BDP02]. An interesting law holding in that papers is that an input, immediately followed by the output of the same message, is equivalent to do nothing. This does not hold in our context. In fact, a receiver of a message cannot re-emit the read message because it is not possible for a service to introduce a message in its own message queue.

Finally, we would to report about related work on the study of services communicating via asynchronous mechanisms and their conversations. In particular, in [FuBS05] the authors present a technique to establish satisfaction of a given property on service conversations from the specifications of the involved services and in [FuBS04] the authors study, given a specification of possible conversations, whether there exists or not a set of services realizing them.

Acknowledgements. We thank the anonymous referees for their comments.

References

[ALM+07] van der Aalst, W.M.P., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: From Public Views to Private Views - Correctness-by-Design for Services. In: Dumas, M., Heckel, R. (eds.) WS-FM 2007. LNCS, vol. 4937, pp. 139–153. Springer, Heidelberg (2008)

[BDP02] Boreale, M., De Nicola, R., Pugliese, R.: Trace and Testing Equivalence on Asynchronous Processes. Information and Computation 172(2), 139–164 (2002)

[BZ07a] Bravetti, M., Zavattaro, G.: Contract based Multi-party Service Composition. In: Arbab, F., Sirjani, M. (eds.) FSEN 2007. LNCS, vol. 4767, pp. 207–222. Springer, Heidelberg (2007)

[BZ07b] Bravetti, M., Zavattaro, G.: Towards a Unifying Theory for Choreography Conformance and Contract Compliance. In: Lumpe, M., Vanderperren, W. (eds.) SC 2007. LNCS, vol. 4829, pp. 34–50. Springer, Heidelberg (2007)

[BZ07c] Bravetti, M., Zavattaro, G.: A Theory for Strong Service Compliance. In: Murphy, A.L., Vitek, J. (eds.) COORDINATION 2007. LNCS, vol. 4467, pp. 96–112. Springer, Heidelberg (2007)

[BZ08] Bravetti, M., Zavattaro, G.: Contract Compliance and Choreography Conformance in the Presence of Message Queues. Technical report, http://www.cs.unibo.it/~bravetti/html/techreports.html

[BGG+05] Busi, N., Gorrieri, R., Guidi, C., Lucchi, R., Zavattaro, G.: Choreography and orchestration: A synergic approach for system design. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 228–240. Springer, Heidelberg (2005)

[CHY07] Carbone, M., Honda, K., Yoshida, N.: Structured Communication-Centred Programming for Web Services. In: De Nicola, R. (ed.) ESOP 2007. LNCS, vol. 4421, pp. 2–17. Springer, Heidelberg (2007)

[CH98] Castellani, I., Hennessy, M.: Testing Theories for Asynchronous Languages. In: Arvind, V., Ramanujam, R. (eds.) FSTTCS 1998. LNCS, vol. 1530, pp. 90–101. Springer, Heidelberg (1998)

[CCL+06] Carpineti, S., Castagna, G., Laneve, C., Padovani, L.: A Formal Account of Contracts for Web Services. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 148–162. Springer, Heidelberg (2006)

[CGP08] Castagna, G., Gesbert, N., Padovani, L.: A Theory of Contracts for Web Services. In: POPL 2008, pp. 261–272. ACM Press, New York (2008)

[DKL+07] Decker, G., Kopp, O., Leymann, F., Weske, M.: BPEL4Chor: Extending BPEL for Modeling Choreographies. In: IEEE 2007 International Conference on Web Services (ICWS), Salt Lake City, Utah, USA. IEEE Copmuter Society, Los Alamitos (2007)

[DeN87] De Nicola, R.: Extensional equivalences for transition systems. Acta Informatica 24(2), 211–237 (1887)

[DH84] De Nicola, R., Hennessy, M.: Testing Equivalences for Processes. Theoretical Computer Science 34, 83–133 (1984)

[FHR+04] Fournet, C., Hoare, C.A.R., Rajamani, S.K., Rehof, J.: Stuck-Free Conformance. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 242–254. Springer, Heidelberg (2004)

[FuBS05] Fu, X., Bultan, T., Su, J.: Synchronizability of Conversations among Web Services. IEEE Trans. Software Eng. 31(12), 1042–1055 (2005)

[FuBS04] Fu, X., Bultan, T., Su, J.: Conversation protocols: a formalism for specification and verification of reactive electronic services. Theor. Comput. Sci. 328(1-2), 19–37 (2004)

[Hoa85] Hoare, T.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs (1985)

[LP07] Laneve, C., Padovani, L.: The must preorder revisited - An algebraic theory for web services contracts. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 212–225. Springer, Heidelberg (2007)

[Mil89] Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)

[RV07] Rensink, A., Vogler, W.: Fair testing. Information and Computation 205(2), 125–198 (2007)

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

[RR02] Rajamani, S.K., Rehof, J.: Conformance Checking for Models of Asynchronous Message Passing Software. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 166–179. Springer, Heidelberg (2002)

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