Choreography and Orchestration Conformance for System Design*

Nadia Busi, Roberto Gorrieri, Claudio Guidi,
Roberto Lucchi, and Gianluigi Zavattaro

Department of Computer Science,
University of Bologna, Italy
{busi, gorrieri, cguidi, lucchi, zavattar}@cs.unibo.it

Abstract. In a previous work we have presented a formal framework devoted to show the relevance of choreography and orchestration in the design of service oriented applications. Even if useful to start a formal investigation of the relationship between choreography and orchestration, the proposed framework was not suitable to specify real case studies. In fact, it simply permitted to specify all possible computations abstracting away from the conditions driving the choice of the actual behaviour. In this paper we tackle this problem by introducing the notion of state variables. The addition of state requires a substantial modification of the entire framework because the same state variable, at the level of choreography, can be actually stored in distributed orchestrators that will need to synchronize in order to maintain consistent views. In order to faithfully investigate this problem we also need to modify the formal model at the orchestration level, moving from synchronous to asynchronous communication as the latter is the communication modality of the ordinary communication infrastructures.

1 Introduction

Choreography and orchestration languages are used for composing service-based applications. The former ones allow to manage applications composed of a number of services in a top view manner, that is the conversation rules which govern the interactions between the services involved in the applications, whereas the latter ones provide a mean to program the internal executable behaviour of some specific service, called orchestrator, responsible to coordinate the collaborating services. These approaches have been separately developed by industrial consortia and international organizations as W3C and OASIS. In particular, WS-CDL [W3C] and WS-BPEL [OAS] specifications represent the most credited languages for the Web Services technology which deal with choreography and orchestration respectively.

Our work aims at synergically exploiting both languages for designing service-based applications where choreography and orchestration can be used for giving different views of the same system. The former one abstracts away from single

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

P. Ciancarini and H. Wiklicky (Eds.): COORDINATION 2006, LNCS 4038, pp. 63–81, 2006.

service peculiarities and, by means of *roles*, describes the behaviours of system participants focusing on their access points and the interactions they perform. The latter one is centered on single services by allowing to design their internal activities by means of workflow operators and message exchange capabilities.

In this context the challenge is to identify the interdependencies between the two views and, in particular, a relationship which allows to verify whether a choreography and an orchestrated system describe the same application. A first effort in this direction has been presented in [BGG+05] where a formal framework, devoted to express the relationship between choreography and orchestration, was introduced. The framework is composed by two calculi, inspired by WS-CDL and WS-BPEL, which capture the peculiarities of choreography and orchestration and a notion of conformance between them. In particular, all the framework is centered on the basic interaction mechanisms and the compositional operators exploited to program more complex patterns. Even if the compositional operators are the same at the choreography and the orchestration levels, the basic interaction mechanisms are significantly different. At the choreography level, the basic interaction mechanisms are atomic synchronizations that permit an instantaneous flow of information between two roles. At the orchestration level, on the contrary, the basic interaction mechanisms consider the act of sending (executed by one process) separated from the act of receiving (executed by another independent process). Finally, conformance is a relationship between the two calculi, inspired to bisimulation, which allows us to verify whether the interactions performed by an orchestrated system behave in accordance with the interactions expressed by a given choreography.

Although the interaction mechanisms that can be described with the framework that we have proposed in [BGG+05] are relevant for managing service-oriented applications, they are insufficient for describing complex systems. The main lack of expressiveness is concerned with the impossibility to describe the choices that are performed depending on the contents of the exchanged messages. For instance, let us consider a electronic shop that allows payments via services provided by either Visa or Master Card depending on the kind of credit card used by the buyer. In this example, the interaction pattern that can be expressed with the choreography calculus in [BGG+05] is an alternative choice between a basic interaction involving the electronic shop and the Visa service, or a basic interaction involving the electronic shop and the Marter Card service. The condition governing this choice cannot be specified. Nevertheless, this condition is definitely relevant when conformance comes into play; for instance, an orchestrator that randomly sends the request for payment to either Visa or Master Card is conformant if we abstract away from this aspect.

The aim of this paper is to tackle this lack of expressiveness of our previous version of the framework. The main idea we follow is to introduce state variables both in the choreography and in the orchestration calculi. This extension requires a substantial redefinition of the entire framework. Intuitively, the main problem is concerned with the fact that a state variable at the choreography level could be distributed among different and possibly distributed processes at

the orchestration level. For instance, an airplane reservation role could be embodied by a travel agency and several airplane company services. In this case, a state variable associated to the airplane reservation role (e.g. the departure date) is distributed among the travel agency and the airplane companies contacted to complete the reservation. Moreover, in order to faithfully model at the orchestration level the problem of synchronizing the distributed views on the shared state variables, we need to consider asynchronous communication[1] as this is the communication modality provided by the ordinary communication infrastructures.

Many technical novelties are necessary to model faithfully state variables; here we simply recall the most relevant ones. In the calculi, choices are now expressed with two distinct operators: an external non-deterministic choice guarded by basic interaction operations and a conditional construct depending on the state of variables. The operational semantics of the orchestration language is strongly influenced by the asynchronous communication mechanism; in fact, in order to model the basic request-response communication pattern, it is necessary to keep track of the relationship between an asynchronous request message and the corresponding asynchronous response message. The main novelties are in the definition of the conformance relation. In particular it deals with the initial internal state of processes and then introduces different kinds of silent actions for distinguishing between internal and coordinating interactions. The former one is used to describe internal synchronization while the latter one expresses the interactions the orchestrators exploit for respecting the constraints of the choreography and that are not considered in it.

There are other works that consider both choreography and orchestration as complementary approaches for managing service oriented systems. In [CHYa] and [CHYb] Honda et al. present two process calculi: one inspired to WS-CDL and the other to pi-calculus, they formalize the two calculi without presenting any formal relation between them. In [DD04] Dijkman and Dumas exploit Petri nets for describing choreography, orchestration and service interface behaviours focusing on the relationship between a single orchestrator w.r.t. a given choreography. In [BBM$^+$05] Schifanella et al., by means of automaton, defines a conformance notion which allows them to test whether interoperability is guaranteed by limiting the notion to systems involving only two peers. Some other papers about conformance exist like [HM05] and [BGJ$^+$05]. The former focuses on automated testing of behavioural contracts provided by a service, whereas the latter deals only with systems composed by two peers.

The paper is structured as follows. Section 2 presents the language for describing choreography whereas in Section 3 we present the orchestration one. In Section 4 the conformance notion is defined and in Section 5 a business application case study is reported. Section 6 concludes the paper by reporting some final remarks and future work.

[1] The formal framework in [BGG$^+$05] considers synchronous communication both for choreography and for orchestration.

2 A Formal Model for Choreography

In this section we introduce the formal model for representing choreography. Intuitively, a choreography is described by three main components: the roles, the initial state constraints on variables and the conversations.

A role represents the behaviour that a participant has to exhibit in order to fulfill the activity defined by the choreography. Each role, which is identified by a name, is equipped with a set of variables and a set of operations.

Operations represent the access point and can have one of the following interaction modalities: *One-Way* or *Request-Response*. Indeed, in WSDL specifications, the most significant types of operations are the *One-Way*, where only the incoming message is defined, and the *Request-Response*, where both the incoming message and the response one are defined.

Let us now introduce the formalization of *roles*, *variables* and *operations*. Let Var be the set of variables ranged over by x, y, z, k. We denote with $\widetilde{x}$ tuples of variables, for instance, we may have $\widetilde{x} = \langle x_1, x_2, ..., x_n \rangle$. Let $OpName$ be the set of operation names, ranged over by o, and $OpType = \{ow, rr\}$ be the set of operation types where ow denotes a One-Way operation whereas rr denotes the Request-Response one. An operation is described by its operation name and operation type. Namely, let $Op = \{(o, t) \mid o \in OpName,\ t \in OpType\}$ be the set of operations where each operation is univocally identified by its name. Let $RName$ be the set of the role names, ranged over by ρ and $Role$, defined as $\{(\rho, \omega, V) \mid \rho \in RName,\ \omega \subseteq Op, V \subseteq Var\}$, be the set which contains all the possible roles.

The state of a choreography describes the variable values and it is represented by a function $\mathcal{S}_C : Var \to Val \cup \{\bot\}$ from variables to the set $Val \cup \{\bot\}$ ranged over by w. Val, ranged over by v, is a generic set of values on which it is defined a total order relation[2]. $\mathcal{S}_C(x)$ represents the value of variable x in the state $\mathcal{S}_C$ ($\mathcal{S}_C(x) = \bot$ means that x is not yet initialized), while $\mathcal{S}_C[v/x]$ denotes the state $\mathcal{S}_C$ where x holds value v (we use $\mathcal{S}_C[\tilde{v}/\tilde{x}]$ when dealing with tuples of variables), formally:

$$\mathcal{S}_C[v/x] = \mathcal{S}'_C \qquad \mathcal{S}'_C(x') = \begin{cases} v & if\ x' = x \\ \mathcal{S}_C(x') & otherwise \end{cases}$$

A choreography can be designed by considering the fact that some variables can hold only a limited set of values within the initial state. This is the case, for example, of a binary variable which can assume only the values 0 or 1. The following grammar allows us to generate logic conditions on variables which we will exploit for expressing the constraints of the initial state and conditional constructs:

$$\chi ::= x \leq e \mid e \leq x \mid \neg\chi \mid \chi \wedge \chi$$

where e denotes an expression which can contain variables references and which can be evaluated into a value v or, when some variables within the expression

[2] we extend such an order relation on the set $Val \cup \{\bot\}$ considering $\bot < v,\ \forall v \in Val$

are not instantiated, into the symbol $\perp$. In the following we use $e \hookrightarrow_{\mathcal{S}_C} w$ to denote that, when the state is $\mathcal{S}_C$, the expression e is evaluated into the value w. It is worth noting that constraints such as $x = v$, $x \neq v$ and $v_1 \leq x < v_2$ can be defined as abbreviations. We exploit the notation $\mathcal{S}_C \vdash \chi$ for denoting that the state $\mathcal{S}_C$ satisfies the condition χ. The satisfaction relation for $\vdash$ is defined by the following rules:

1. $\mathcal{S}_C(x) = \perp \Rightarrow \mathcal{S}_C \vdash (x \leq \perp \wedge \perp \leq x)$
2. $e \hookrightarrow_{\mathcal{S}_C} v, \mathcal{S}_C(x) \leq v \Rightarrow \mathcal{S}_C \vdash x \leq e$
3. $e \hookrightarrow_{\mathcal{S}_C} v, v \leq \mathcal{S}_C(x) \Rightarrow \mathcal{S}_C \vdash e \leq x$
4. $\mathcal{S}_C \vdash \chi' \wedge \mathcal{S}_C \vdash \chi'' \Rightarrow \mathcal{S}_C \vdash \chi' \wedge \chi''$
5. $\neg(\mathcal{S}_C \vdash \chi) \Rightarrow \mathcal{S}_C \vdash \neg\chi$

We highlight the fact that rule 1 states that when a variable x is defined with value $\perp$ the only condition which can be satisfied on such a state is $x = \perp$.

The conversations among the roles are defined by using a conversation language whose definition follows where we intend I as a finite non-empty subset of natural numbers:

$$C ::= \mathbf{0} \mid \eta \mid C; C \mid C|C \mid \sum_{i\in I}^{+} \eta_i; C_i \mid \sum_{i\in I}^{\oplus} \chi_i ? \eta_i; C_i$$
$$\eta ::= (\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, dir) \mid x := e$$

In the following we use CL_P, ranged over by C, to denote the set of conversations. η represents the basic building block of a conversation which can be an interaction or an assignment. $(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, dir)$ means that an interaction from role ρ_A to role ρ_B is performed. In particular, o is the name of the operation $(o, t) \in Op$ on which the message exchange is performed. Variables $\widetilde{x}$ and $\widetilde{y}$ are those used by the sender and the receiver, respectively and $dir \in \{\uparrow, \downarrow\}$ represents if the interaction is a request ($\uparrow$) or a response ($\downarrow$) one. $x := e$ means that the result of the evaluation of the expression e is assigned to the variable x. Coherently with the grammar of logic conditions, here we abstract away from the syntax of expression e and we exploit the evaluation function $\hookrightarrow_{SC}$ introduced above. A conversation can be the null one ($\mathbf{0}$), a basic operation (η), the sequential composition ($C; C$), the parallel composition ($C \mid C$) or two different kind of choices: the non-deterministic choice ($\sum_{i\in I}^{+} \eta_i; C_i$) and the deterministic one ($\sum_{i\in I}^{\oplus} \chi_i ? \eta_i; C_i$). The former non-deterministically selects a conversation to execute independently from the state of the choreography whereas in the latter the selection is driven by guard conditions χ. The choice is deterministic because the guards are evaluated in a sequential order.

The semantics of CL_P is defined in terms of a labelled transition system [Kel76] which describes the evolution of a conversation joined with a state. Let $Act_C = \{\mu \mid \mu = (\rho_A, \rho_B, o, \tilde{v}, dir)\} \cup \{\tau\}$ be the set of actions ranged over by ν where μ represents parameterized interactions. $(C, \mathcal{S}_C) \xrightarrow{\nu} (C', \mathcal{S}'_C)$ means that the conversation C in the state $\mathcal{S}_C$ evolves in one step in a configuration $(C', \mathcal{S}'_C)$ performing the action ν. Let Γ_C be the set of all possible states over the variables in Var. We define $\rightarrow \subseteq (CL_P, \Gamma_C) \times Act_C \times (CL_P, \Gamma_C)$ as the least relation which

Table 1. Semantics of CL_P

(Interaction 1)
$$((\rho_A, \rho_B, o, \tilde{x}, \tilde{y}, \uparrow), \mathcal{S}_C) \xrightarrow{\mu} (\mathbf{0}, \mathcal{S}_C[\tilde{w}/\tilde{y}]),\ \mu = (\rho_A, \rho_B, o, \tilde{w}, \uparrow),\ \tilde{w} = \mathcal{S}_C(\tilde{x})$$

(Interaction 2)
$$((\rho_A, \rho_B, o, \tilde{x}, \tilde{y}, \downarrow), \mathcal{S}_C) \xrightarrow{\mu} (\mathbf{0}, \mathcal{S}_C[\tilde{w}/\tilde{x}]),\ \mu = (\rho_A, \rho_B, o, \tilde{w}, \downarrow),\ \tilde{w} = \mathcal{S}_C(\tilde{y})$$

(Assign)
$$\frac{e \hookrightarrow_{\mathcal{S}_C} v}{(x := e, \mathcal{S}_C) \xrightarrow{\tau} (\mathbf{0}, \mathcal{S}_C[v/x])} \qquad \text{(Sequence)}\ \frac{(C, \mathcal{S}_C) \xrightarrow{\nu} (C', \mathcal{S}'_C)}{(C; D, \mathcal{S}_C) \xrightarrow{\nu} (C'; D, \mathcal{S}'_C)}$$

(Parallel)
$$\frac{(C, \mathcal{S}_C) \xrightarrow{\nu} (C', \mathcal{S}'_C)}{(C \mid D, \mathcal{S}_C) \xrightarrow{\nu} (C' \mid D, \mathcal{S}'_C)} \qquad \text{(Congr)}\ \frac{C' \equiv C, (C, \mathcal{S}_C) \xrightarrow{\nu} (D, \mathcal{S}'_C), D \equiv D'}{(C', \mathcal{S}_C) \xrightarrow{\nu} (D', \mathcal{S}'_C)}$$

(Choice 1)
$$\frac{(\eta_i; C_i, \mathcal{S}_C) \xrightarrow{\nu} (C'_i, \mathcal{S}'_C), i \in I}{(\sum^{+}_{i \in I} \eta_i; C_i, \mathcal{S}_C) \xrightarrow{\nu} (C'_i, \mathcal{S}'_C)} \qquad \text{(Choice 2)}\ \frac{\mathcal{S}_C \vdash \chi_i, (\eta_i, \mathcal{S}_C) \xrightarrow{\nu} (\mathbf{0}, \mathcal{S}'_C), \mathcal{S}_C \not\vdash \chi_j, j \in I, j < i}{(\sum^{\oplus}_{i \in I} \chi_i ? \eta_i; C_i, \mathcal{S}_C) \xrightarrow{\nu} (C_i, \mathcal{S}'_C)}$$

(Structural Congruenge)
$$\mathbf{0}; C \equiv C \qquad C \mid \mathbf{0} \equiv C$$
$$C \mid D \equiv D \mid C \qquad (C \mid D) \mid F \equiv C \mid (D \mid F)$$

satisfies the axioms and rules of Table 1 and closed w.r.t. $\equiv$, where $\equiv$ is the least congruence relation satisfying the axioms at the end of Table 1.

The structural congruence $\equiv$, which equates the conversations whose behaviour cannot be distinguished, expresses that $(C, \mid)$ is an abelian monoid where $\mathbf{0}$ is the null element. Furthermore, the rule $\mathbf{0}; C \equiv C$ means that when a conversation completes then the other one which follows in sequence can be performed.

The description of axioms and rules follows. The axioms Interaction describe that an interaction, which is a request or a response one depending on the value of *dir*, is performed. When a request is performed ($dir = \uparrow$) the information contained in the variables $\widetilde{x}$ within the sender role ρ_A are passed to the variables $\widetilde{y}$ within the receiver role ρ_B exploiting the operation o of the role ρ_B. When a response is performed ($dir = \downarrow$) the information contained in the variables $\widetilde{y}$ within the receiver role ρ_B are passed to the variables $\widetilde{x}$ within the sender role ρ_A exploiting the operation o of the role ρ_B. The rule Assign states that the resulting value of the expression e, evaluated within the state $\mathcal{S}_C$, is assigned to a variable x thus updating the state of the choreography. Rules Sequence, Parallel and Congr are standard. Rule Choice 1 deals with the non-deterministic choice which is independent of the state. It is worth noting that the construct $\eta_i; C_i$ guarantees that the conversation can always move. Rule Choice 2, on the contrary, deals with deterministic choice depending on the state $\mathcal{S}_C$. Here, we want to highlight that guards are sequentially evaluated and the first which is satisfied in the state $\mathcal{S}_C$ is selected.

Now we are ready to define a choreography. A choreography, denoted by $\mathcal{C}$, is defined by the tuple (C, Σ, X) where $C \in CL_P$, $\Sigma \subseteq Role$ is a finite set containing all involved roles and X is a logic condition which expresses the variables constraints of the initial state. The constraint expressed by X is strictly pertaining to the choreography because it expresses the set of possible values that variables can hold at the initial state. It is worth noting that such a constraint is not considered when the system evolves.

We say that a choreography $\mathcal{C} = (C, \Sigma, X)$ is *well-formed* if: i) the sets of variables used by roles are disjoint, ii) the variables appearing in each guard condition and each assignment in C involve variables of a single role. In the following we consider only well-formed choreographies.

3 A Formal Model for Orchestration

An orchestrator can be seen as a process, associated to an identifier, that can exchange information, represented by variables, with other processes. Let ID be the set of possible orchestrator identifiers ranged over by id. The language is defined as it follows where we intend I as a finite non-empty set of indexes:

$$P ::= \mathbf{0} \mid \bar{o} \mid \bar{o}(\tilde{y}) \mid \bar{o}(\tilde{x}, \tilde{y}) \mid \epsilon \mid x := e \mid P; P \mid P \mid P \mid \sum^{+}_{i \in I} \epsilon_i; P_i \mid \sum^{\oplus}_{i \in I} \chi_i ? P_i$$
$$\epsilon ::= o \mid o(\tilde{x}) \mid o(\tilde{x}, \tilde{y}, P)$$
$$E ::= [P, \mathcal{S}]_{id} \mid E \parallel E$$

An orchestrated system E consists of the parallel composition of orchestrators. An orchestrator $[P, \mathcal{S}]_{id}$ is a process P identified by id whose variables state is $\mathcal{S}$. The variables state of an orchestrator is described by a function $\mathcal{S} : Var \rightarrow Val \cup \{\perp\}$ mapping variables to values as defined for the choreography. Informally, the idea is that orchestrators are executed on different locations, thus they can be composed by using only the parallel operator ($\parallel$). Processes can be composed in parallel ($|$), sequence ($;$) and with two different alternative composition operators: one is composed of input guarded processes and the other one is composed of processes guarded by conditions on variables state (such processes are of the form $\chi ? P$ where χ is the condition associated to P).

0 represents the null process. Communication mechanisms model Web Services One-Way and Request-Response operations. In particular, we have three kinds of primitives for synchronization, one for the internal synchronization and two for the external one. The former simply consists of a channel o that different threads of the process running in parallel can use to coordinate their activities. In this case no message is exchanged; this is because the orchestrator variables are shared by all threads running on that orchestrator. The primitives for external synchronization, that is between different orchestrators, are the following ones: $o(\tilde{x})$ and $\bar{o}(\tilde{y})$ represent the input and the output of a single message whereas the primitives $o(\tilde{x}, \tilde{y}, P)$ and $\bar{o}(\tilde{x}, \tilde{y})$ represent coupled messages exchanges. In particular we have that $o(\tilde{x})$ represents a One-Way operation whose name is o where the received information is stored in the tuple of variable $\tilde{x}$ of the receiver.

Table 2. Axioms over P

$$\text{(In)}\quad (o,\mathcal{S}) \xrightarrow{o} (\mathbf{0},\mathcal{S}) \qquad \text{(Out)}\quad (\bar{o},\mathcal{S}) \xrightarrow{\bar{o}} (\mathbf{0},\mathcal{S})$$

$$\text{(One-WayOutAsyn)}\quad (\bar{o}(\tilde{y}),\mathcal{S}) \xrightarrow{\tau} (\langle \bar{o}(\mathcal{S}(\tilde{y}))\rangle,\mathcal{S}) \qquad \text{(One-WayOut)}\quad (\langle \bar{o}(\tilde{v})\rangle,\mathcal{S}) \xrightarrow{\bar{o}(\tilde{v})} (\mathbf{0},\mathcal{S}) \qquad \text{(One-WayIn)}\quad (o(\tilde{x}),\mathcal{S}) \xrightarrow{o(\tilde{v})} (\mathbf{0},\mathcal{S}[\tilde{v}/\tilde{x}])$$

$$\text{(Req-OutAsyn)}\quad (\bar{o}(\tilde{x},\tilde{y}),\mathcal{S}) \xrightarrow{\tau} (\langle \bar{o}(\mathcal{S}(\tilde{x}),\tilde{y})\rangle,\mathcal{S}) \qquad \text{(Req-Out)}\quad (\langle \bar{o}(\tilde{v},\tilde{y})\rangle,\mathcal{S}) \xrightarrow{\bar{o}(\tilde{v},\tilde{y})(n)} (o_n(\tilde{y}),\mathcal{S})$$

$$\text{(Resp-OutAsyn)}\quad (\bar{o}_n(\tilde{y}),\mathcal{S}) \xrightarrow{\tau} (\langle \bar{o}_n(\mathcal{S}(\tilde{y}))\rangle,\mathcal{S}) \qquad \text{(Resp-Out)}\quad (\langle \bar{o}_n(\tilde{v})\rangle,\mathcal{S}) \xrightarrow{\bar{o}_n(\tilde{v})} (\mathbf{0},\mathcal{S})$$

$$\text{(Req-In)}\quad (o(\tilde{x},\tilde{y},P),\mathcal{S}) \xrightarrow{o(\tilde{v},\tilde{y})(n)} (P;\bar{o}_n(\tilde{y}),\mathcal{S}[\tilde{v}/\tilde{x}]) \qquad \text{(Resp-In)}\quad (o_n(\tilde{x}),\mathcal{S}) \xrightarrow{o_n(\tilde{v})} (\mathbf{0},\mathcal{S}[\tilde{v}/\tilde{x}])$$

$\bar{o}(\tilde{y})$ represents a One-Way invocation whose name is o and the sent information is stored in the tuple $\tilde{y}$ of the sender. $o(\tilde{x},\tilde{y},P)$ represents a Request-Response operation whose name is o. In this case the process receives a message and stores the received information in $\tilde{x}$ then it executes the process P and, at the end, sends the information contained in $\tilde{y}$ as a response message to the invoker. Finally, $\bar{o}(\tilde{x},\tilde{y})$ represents the invocation of a Request-Response operation whose name is o. The process sends the information contained in $\tilde{x}$ as a request message and stores the information of the response message in $\tilde{y}$. The processes $x := e$ deal with variable assignment.

Let OL be the set of all the orchestrated system ranged over by E. The semantics of OL is defined in terms of a labelled transition system which describes the evolution of an orchestrated system. To this end we exploit the syntax of the language enhanced with terms we use to describe asynchronous communication as it follows:

$$P ::= \ldots \mid \langle\overline{o}(\tilde{v})\rangle \mid \langle\overline{o}(\tilde{v},\tilde{y})\rangle \mid \langle\bar{o}_n(\tilde{v})\rangle$$

We define $\rightarrow$ as the least relation which satisfies the axioms and rules of Tables 2, 3 and 4. Let $Act_{OL} = \{\bar{o}, o, \bar{o}(\tilde{v}), o(\tilde{v}), \bar{o}(\tilde{v},\tilde{k})(n), o(\tilde{v},\tilde{z})(n), \bar{o}_n(\tilde{v}), o_n(\tilde{v}), \sigma, \tau\}$ be the set of actions ranged over by γ. σ is a parameterized action of the form $(id, id', o, \tilde{v}, dir)$ where id, id' are orchestrators ids, o is an operation name, $\tilde{v}$ are tuples of values and $dir \in \{\uparrow,\downarrow\}$.

Table 2 deals with In, Out, One-WayOut, One-Way-In, Req-Out, Req-In, Resp-Out, Resp-In axioms. It is worth noting that the processes $\langle\overline{o}\rangle$, $\langle\overline{o}(\tilde{y})\rangle$,$\langle\overline{o}(\tilde{x},\tilde{y})\rangle$,$\langle\bar{o}_n(\tilde{v})\rangle$ model asynchronous communication which characterize Service Oriented Computing and in particular Web Services. Indeed, every time an outcoming message process is performed, they "freeze" the value of the variable to be sent and, by exploiting the structural congruence rules of Table 3, they

Table 3. Rules over P

$$\text{(Assign)}\ \frac{e \hookrightarrow_{\mathcal{S}} v}{(x := e, \mathcal{S}) \xrightarrow{\tau} (\mathbf{0}, \mathcal{S}[v/x])} \qquad \text{(Int-Sync)}\ \frac{(P, \mathcal{S}) \xrightarrow{o} (P', \mathcal{S})\ ,\ (Q, \mathcal{S}) \xrightarrow{\bar{o}} (Q', \mathcal{S})}{(P \mid Q, \mathcal{S}) \xrightarrow{\tau} (P' \mid Q', \mathcal{S})}$$

$$\text{(CongrP)}\ \frac{P \equiv_P P'\ ,\ (P', \mathcal{S}) \xrightarrow{\gamma} (Q', \mathcal{S}'),\ Q' \equiv_P Q}{(P, \mathcal{S}) \xrightarrow{\gamma} (Q, \mathcal{S}')}$$

$$\text{(Par-Int)}\ \frac{(P, \mathcal{S}) \xrightarrow{\gamma} (P', \mathcal{S}')}{(P \mid Q, \mathcal{S}) \xrightarrow{\gamma} (P' \mid Q, \mathcal{S}')} \qquad \text{(Seq)}\ \frac{(P, \mathcal{S}) \xrightarrow{\gamma} (P', \mathcal{S}')}{(P; Q, \mathcal{S}) \xrightarrow{\gamma} (P'; Q, \mathcal{S}')}$$

$$\text{(Choice 1)}\ \frac{(\epsilon_i; P_i, \mathcal{S}) \xrightarrow{\gamma} (P', \mathcal{S}') \quad i \in I}{(\sum^{+}_{i \in I} \epsilon_i; P_i, \mathcal{S}) \xrightarrow{\gamma} (P', \mathcal{S}')} \qquad \text{(Choice 2)}\ \frac{\mathcal{S} \vdash \chi_i \quad \mathcal{S} \nvdash \chi_j, j \in I, j < i}{(\sum^{\oplus}_{i \in I} \chi_i ? P_i, \mathcal{S}) \xrightarrow{\tau} (P_i, \mathcal{S})}$$

(Structural Congruenge over P)

$$P \mid \mathbf{0} \equiv_P P \quad \mathbf{0}; P \equiv_P P \quad (P \mid Q) \equiv_P (Q \mid P) \quad (P \mid Q) \mid R \equiv_P P \mid (Q \mid R)$$
$$(\langle \bar{o} \rangle ; Q) \equiv_P (\langle \bar{o} \rangle \mid Q) \qquad (\langle \bar{o}(\tilde{v}) \rangle ; Q) \equiv_P (\langle \bar{o}(\tilde{v}) \rangle \mid Q)$$
$$(\langle \bar{o}(\tilde{v}, \tilde{y}) \rangle ; Q) \equiv_P (\langle \bar{o}(\tilde{v}, \tilde{y}) \rangle \mid Q) \qquad (\langle \bar{o}_n(\tilde{v}) \rangle ; Q) \equiv_P (\langle \bar{o}_n(\tilde{v}) \rangle \mid Q)$$

goes in parallel with the other processes. In the Req-In rule, after the reception of a request on a Request-Response operation the process P must be executed before sending the response.

In Table 3 there are the rules over P where the Assign one deals with variables assignment within the orchestrators. Rule Int-Sync deals with internal synchronization whereas CongrP with internal structural congruence denoted by $\equiv_P$. Par-Int and Seq describe the behaviour of processes composed in parallel and sequentially, respectively. Finally Choice1 and Choice2 describe the behavior of the two alternative composition operators. The former one nondeterministically selects among the processes guarded by inputs which can be consumed, while the latter one resembles the deterministic choice, depending on variables state, used in choreography.

In Table 4 the rules at the level of orchestrator system are considered. Rule One-WaySync deals with the synchronization on a One-Way operation between two orchestrators whereas the rules Req-Sync and Resp-Sync deal with that on a Request-Response one. Rule Req-Sync exploits a fresh label n which is generated in order to univocally link the response synchronization defined in rule Resp-Sync. Considering the axioms Req-Out and Req-In indeed, the Request-Response primitives will be transformed into two One-Way (invocation and reception) identified by the label n which is unique and univocally

Table 4. Rules over E

(Rules over E)

(One-WaySync)
$$\frac{[P,\mathcal{S}]_{id} \xrightarrow{\bar{o}(\tilde{v})} [P',\mathcal{S}']_{id} \ , \ [Q,\mathcal{T}]_{id'} \xrightarrow{o(\tilde{v})} [Q',\mathcal{T}']_{id'} \ , \sigma = (id, id', o, \tilde{v}, \uparrow)}{[P,\mathcal{S}]_{id} \parallel [Q,\mathcal{T}]_{id'} \xrightarrow{\sigma} [P',\mathcal{S}']_{id} \parallel [Q',\mathcal{T}']_{id'}}$$

(Req-Sync)
$$\frac{[P,\mathcal{S}]_{id} \xrightarrow{\bar{o}(\tilde{v},\tilde{y})(n)} [P',\mathcal{S}']_{id} \ , \ [Q,\mathcal{T}]_{id'} \xrightarrow{o(\tilde{v},\tilde{y})(n)} [Q',\mathcal{T}']_{id'} \ , \ n \ fresh \ , \ \sigma = (id, id', o, \tilde{v}, \uparrow)}{[P,\mathcal{S}]_{id} \parallel [Q,\mathcal{T}]_{id'} \xrightarrow{\sigma} [P',\mathcal{S}']_{id} \parallel [Q',\mathcal{T}']_{id'}}$$

(Resp-Sync)
$$\frac{[P,\mathcal{S}]_{id} \xrightarrow{o_n(\tilde{v})} [P',\mathcal{S}']_{id} \ , \ [Q,\mathcal{T}]_{id'} \xrightarrow{\bar{o}_n(\tilde{v})} [Q',\mathcal{T}']_{id'} \ , \sigma = (id, id', o, \tilde{v}, \downarrow)}{[P,\mathcal{S}]_{id} \parallel [Q,\mathcal{T}]_{id'} \xrightarrow{\sigma} [P',\mathcal{S}']_{id} \parallel [Q',\mathcal{T}']_{id'}}$$

(Par-Ext)
$$\frac{E_1 \xrightarrow{\gamma} E_1'}{E_1 \parallel E_2 \xrightarrow{\gamma} E_1' \parallel E_2}$$

(CongrE)
$$\frac{E_1 \equiv E_1' \ , \ E_1' \xrightarrow{\gamma} E_2' , \ E_2' \equiv E_2}{E_1 \xrightarrow{\gamma} E_2}$$

(Int-Ext)
$$\frac{(P,\mathcal{S}) \xrightarrow{\gamma} (P',\mathcal{S}')}{[P,\mathcal{S}]_{id} \xrightarrow{\gamma} [P',\mathcal{S}']_{id}}$$

(Structural Congruence over E)
$$\frac{P \equiv_P Q}{[P,\mathcal{S}]_{id} \equiv [Q,\mathcal{S}]_{id}} \qquad E_1 \parallel E_2 \equiv E_2 \parallel E_1 \qquad E_1 \parallel (E_2 \parallel E_3) \equiv (E_1 \parallel E_2) \parallel E_3$$

determined during the synchronization. It is worth noting that all the synchronizations which are performed between different orchestrators are labelled with an action σ. This fact will be fundamental for the definition of the conformance notion presented in the next section. Par-Ext deals with external parallel composition and CongrE is for external structural congruence denoted by $\equiv$. Int-Ext expresses the fact that an orchestrator behaves in accordance with its internal processes.

4 Conformance Between Choreography and Orchestration

Our proposal defines a conformance notion based on a relation between the labelled transition system of choreography and the labelled transition system of another model obtained from the orchestration system by associating choreography roles to the orchestrators. In particular, let $\mathcal{C} = (C, \Sigma, X)$ be a choreography and E be an orchestrated system. We define a *joining function*, named Ψ, for associating the orchestrators and the variables of E to the roles of $\mathcal{C}$ and we test the conformance, up to Ψ, of E and $\mathcal{C}$ by using a relation where the σ labels of the former are compared with the μ ones of the latter.

Definition 1 (joining functions). *A joining function is an element of the set*

$$\{\Psi \mid \Psi : ID \to RName \cup \{\bot\} \times (Var \to Var \cup \{\bot\})\}$$

containing functions which associate to each orchestrator identifier a pair composed of a choreography role (or the $\bot$ value in case no role is associated) and a function from orchestrator variables to choreography variables (or the $\bot$ value in case no variable is associated). We denote with Ψ^1 the projection of Ψ on the first element of the pair (the associated role), with Ψ^2 the projection on the second element (the variable mapping function).

Given a joining function Ψ and an action $\sigma = (id, id', o, \tilde{v}, dir)$ of a given orchestrated system where id and id' are orchestrator identifiers, o is an operation, $\tilde{v}$ are tuples of values and $dir \in \{\uparrow, \downarrow\}$, we denote with

$$\Psi[\sigma] = (\Psi^1(id), \Psi^1(id'), o, \tilde{v}, dir)$$

the renaming of the orchestrator identifiers with the joined roles. The projection Ψ^2 will be exploited for joining the initial values of the choreography variables to the related ones of the orchestrated system.

Now we introduce the *conformance notion* between a choreography and an orchestrated system which exploits a relation, named *conformability relation*, inspired to bisimulation [Mil89]. Given a choreography $\mathcal{C} = (C, \Sigma, X)$ an orchestrated system E and a joining function Ψ, the idea is to consider all the possible choreography states which satisfy the initial constraint X and for each of them test, up to Ψ, the conformability ($\triangleright_\Psi$) between the labelled transition system of the choreography and a new labeled transition system for the orchestrated systems that we call the *joined labelled transition system*. In particular, in this new transition system the initial values of the variables of the orchestrated system are joined with the choreography ones up to Ψ^2. Furthermore, some hiding operators are applied to the orchestrated system in order to observe only those interactions which are relevant for the choreography. Hiding consists of replacing labels with τ or with a special label $\tilde{\tau}$ and it is applied to three kinds of actions: (i) the interactions that involve an orchestrator not joined with any role are replaced with $\tilde{\tau}$; (ii) the interactions performed on operations not declared in the choreography are replaced with $\tilde{\tau}$; (iii) the interactions which are performed between orchestrators joined with the same role are replaced with τ. The case (iii) is concerned with interactions that corresponds to internal operation within the same role at the level of choreography, while cases (i) and (ii) correspond to coordinating actions that are introduced only at the level of orchestration in order to coordinate distributed orchestrators. For this reason we distinguish these two kinds of actions introducing the new label $\tilde{\tau}$. Formally, such a difference comes into play in the conformability relation.

Definition 2 (Joined labelled transition system). *Given a choreography $\mathcal{C} = (C, \Sigma, X)$, an orchestrated system $E \in OL$ and a joining function Ψ such that $Im^1(\Psi) = \Sigma \cup \{\bot\}$[3], let ω_C be the set of operations involved within the*

[3] $Im^1(\Psi) = \{\Psi^1(id) \mid id \in ID\}$

choreography $\mathcal{C}$, let ω_o be the set of operations exhibited by the processes of E and let $E_{OP} = \omega_o/\omega_C$ be the set of operations exhibited by E and which do not appear within the roles of $\mathcal{C}$. Let $E_\perp$ be the set of orchestrator identifiers id of E for which $\Psi(id) = \perp$. We denote the joined labelled transition system with:

$$E^\frown\Psi^2/E_{OP}/\!\!/E_\perp/\!\!/\!/E_{id}$$

where:

- *$E^\frown\Psi^2$ is an operator which associates the values of the choreography variables in $\mathcal{S}_C$ to the corresponding variables in the states of E up to the joining function Ψ^2. Formally let $\tilde{x}_{id}$ and $\tilde{y}_{id}$ be the tuples of variables in Var which belong to the state of the orchestrator id and for which the following conditions hold respectively: $\Psi^2(id)(\tilde{x}_{id}) \neq \perp$, $\Psi^2(id)(\tilde{y}_{id}) = \perp$ and let $\tilde{v}_{id}$ be the tuple of values of the choreography variables joined with the variables $\tilde{x}_{id}$ that is $\tilde{v}_{id} = \mathcal{S}_C(\Psi^2(id)(\tilde{x}_{id}))$. We have that the $E^\frown\Psi^2$ is inductively defined as follows:*

  - $[P, \mathcal{S}]^\frown_{id}\Psi^2 = [P, \mathcal{S}[\tilde{v}_{id}/\tilde{x}_{id}, \perp/\tilde{y}_{id}]]_{id}$
  - $E^\frown\Psi^2 = [P, \mathcal{S}[\tilde{v}_{id}/\tilde{x}_{id}, \perp/\tilde{y}_{id}]]_{id} \parallel E'^\frown\Psi^2$

- *$/E_{OP}$ is a hiding operator which hides, replacing with $\tilde{\tau}$ moves, all the transitions which contain operations contained in E_{OP}*
- *$/\!/E_\perp$ is a hiding operator which hides, replacing with $\tilde{\tau}$ moves, all the transitions which contain orchestrators not joined with any role.*
- *$/\!/\!/E_{id}$ hides, replacing with τ moves, all the interactions between the same role (the id of the sender is the same of the receiver).*

In the following we present the conformability relation between the labelled transition system of the choreography and the orchestrated system one. Conformability is inspired by bisimulation but some differences exist. In particular, conformability considers $\tilde{\tau}$ and τ moves differently. Indeed, in order to abstract away from coordinating interactions on the orchestration side, we will exploit the particular arrows $\stackrel{\sigma}{\Rightarrow}$ and $\stackrel{\tau}{\Rightarrow}$ for representing the concatenation of the following transitions $\stackrel{\tilde{\tau}^*}{\rightarrow}\stackrel{\sigma}{\rightarrow}\stackrel{\tilde{\tau}^*}{\rightarrow}$ and $\stackrel{\tilde{\tau}^*}{\rightarrow}\stackrel{\tau}{\rightarrow}\stackrel{\tilde{\tau}^*}{\rightarrow}$ respectively. This means that we focus on observable interactions which are represented by σ for the orchestrated system and by $\mu = \Psi^1[\sigma]$ for the choreography. Furthermore, τ actions in choreography, which correspond to assignments, are related to internal role actions in orchestration. It is worth noting that here, we are not interested to distinguish between deadlock and termination states both in choreography and orchestration which are related in conformability by introducing the set of states $C_\delta(\mathcal{S}_C)$ and E_δ defined in the following.

Definition 3 (Conformability). *Let Ψ be a joining function. A relation $\mathcal{R}_\Psi \subseteq ((CL_P, \Gamma_C) \times OL)$ is a conformability relation if $((C, \mathcal{S}_C), E) \in \mathcal{R}_\Psi$ implies that $C \in C_\delta(\mathcal{S}_C)$ and $E \in E_\delta$ or, for all $\mu \in Act_C$ and for all $\sigma \in Act_{OL}$, the following conditions hold:*

1. $(C, \mathcal{S}_C) \xrightarrow{\mu} (C', \mathcal{S}'_C) \Rightarrow E \xrightarrow{\hat{\tau}}{}^{*} E' \wedge E' \xrightarrow{\sigma} E'' \wedge E'' \xrightarrow{\hat{\tau}}{}^{*} E'''$
$\wedge ((C', \mathcal{S}'_C), E''') \in \mathcal{R}_\Psi \wedge \Psi^1[\sigma] = \mu$

2. $(C, \mathcal{S}_C) \xrightarrow{\tau} (C', \mathcal{S}'_C) \Rightarrow E \xrightarrow{\hat{\tau}}{}^{*} E' \wedge E' \xrightarrow{\tau} E'' \wedge E'' \xrightarrow{\hat{\tau}}{}^{*} E'''$
$\wedge ((C', \mathcal{S}'_C), E''') \in \mathcal{R}_\Psi$

3. $E \stackrel{\sigma}{\Rightarrow} E' \Rightarrow (C, \mathcal{S}_C) \xrightarrow{\tau}{}^{*} (C', \mathcal{S}'_C) \wedge (C', \mathcal{S}'_C) \xrightarrow{\mu} (C'', \mathcal{S}''_C) \wedge$
$\wedge (C'', \mathcal{S}''_C) \xrightarrow{\tau}{}^{*} (C''', \mathcal{S}'''_C) \wedge ((C''', \mathcal{S}'''_C), E') \in \mathcal{R}_\Psi \wedge \Psi^1[\sigma] = \mu$

4. $E \stackrel{\tau}{\Rightarrow} E' \Rightarrow ((C, \mathcal{S}_C), E') \in \mathcal{R}_\Psi \vee ((C, \mathcal{S}_C) \xrightarrow{\tau} (C', \mathcal{S}'_C) \wedge ((C', \mathcal{S}'_C), E') \in \mathcal{R}_\Psi)$

where $\hat{\tau} \in \{\tau, \tilde{\tau}\}$*, the arrow* $\stackrel{\gamma}{\Rightarrow}$ *means the concatenation of the following transitions:* $\xrightarrow{\tilde{\tau}^*}\xrightarrow{\gamma}\xrightarrow{\tilde{\tau}^*}$ *and* $C_\delta(\mathcal{S}_C)$ *and* E_δ *are defined as follows:*

- $C_\delta(\mathcal{S}_C) = \{C \in CL_P \mid \forall C' \in CL_P \nexists \nu, \mathcal{S}'_C \text{ s.t. } (C, \mathcal{S}_C) \xrightarrow{\nu} (C', \mathcal{S}'_C)\}$
- $E_\delta = \{E \in OL \mid \nexists E' \in OL \text{ s.t. } E \xrightarrow{\sigma} E' \vee E \xrightarrow{\tilde{\tau}} E' \vee E \xrightarrow{\tau} E'\}$

We write $(C, \mathcal{S}_C) \triangleright_\Psi E$ *if there exists a conformability relation* $\mathcal{R}_\Psi$ *such that* $((C, \mathcal{S}_C), E) \in \mathcal{R}_\Psi$*.*

We can now conclude this section reporting the formal definition of our conformance notion.

Definition 4 (Conformance). *Given a choreography* $\mathcal{C} = (C, \Sigma, X)$*, an orchestrated system* $E \in OL$ *and a joining function* Ψ *such that* $Im^1(\Psi) = \Sigma \cup \{\bot\}$*, let* ω_C *be the set of operations involved within the choreography* $\mathcal{C}$*, let* ω_o *be the set of operations exhibited by the processes of* E *and let* $E_{OP} = \omega_o / \omega_C$ *be the set of operations exhibited by* E *and which do not appear within the roles of* $\mathcal{C}$*. Let* $E_\bot$ *be the set of orchestrator identifiers id of* E *for which* $\Psi(id) = \bot$*. We say that* E *is conformant to* $\mathcal{C}$ *if the following condition holds:*

$$\forall \mathcal{S}_C \in \Gamma_C \text{ s.t. } \mathcal{S}_C \vdash X, (C, \mathcal{S}_C) \triangleright_\Psi E^\frown \Psi^2 / E_{OP} /\!\!/ E_\bot /\!\!/\!/ E_{id}$$

Observe that on the right hand side of $\triangleright_\Psi$ *the joined labelled transition system of the orchestrated system defined in Definition 2 is considered.*

5 Example

Here we reason about the meaning of conformance by using an example. Let us now consider a business scenario where a customer invokes a market service in order to buy some goods and it receives the price as a response. Considering the price, the customer will buy or not the goods (in this case, for the sake of clarity, we have choosen 100 as a constant for discriminating the price but, in order to abstract away from this value, it could be possible to use a variable with a range of values). If the customer sends a message for buying the market will invoke

a supplier service for making the order. The supplier service will accept or not the order. In the case the order can be fulfilled, the market service will invoke a bank service for the payment and will return a positive answer to the customer, the bank service concurrently will send a receipt to the cutomer.

In order to define the choreography let us consider four roles: ρ_C which represents the customer behaviour, ρ_M which represents the market service, ρ_B which represents the bank service for credit card payment and ρ_S which represents the supplier service. For each role we define the following operations and sets of variables:

$$\omega_C = \{(\text{RESULT}, ow), (\text{RECEIPT}, ow)\},\ \ \omega_M = \{(\text{PRICE}, rr), (\text{BUY}, ow)\},$$
$$\omega_S = \{(\text{ORDER}, rr)\},\ \ \omega_B = \{(\text{PAY}, ow)\}.$$
$$V_C = \{good_C, num_C, buy_C, card_C, ncard_C, price_C, outcome_C, receipt_C\}$$
$$V_M = \{good_M, num_M, buy_M, card_M, ncard_M, price_M, outcome_M\}$$
$$V_S = \{good_S, num_S, price_S, outcome_S\}$$
$$V_B = \{card_B, ncard_B, receipt_B, price_B\}$$

Let Σ be the set of roles defined in the following way:
$\Sigma = \{(\rho_C, \omega_C, V_C), (\rho_M, \omega_M, V_M), (\rho_S, \omega_S, V_S), (\rho_B, \omega_B, V_B)\}$.

Let Con be the following conversation:

$$\begin{aligned} Con ::= {} & (PriceReq; BuyReq; (buy_C = accepted?Order; BuyResp)) \mid \\ \mid {} & outcome_M = OK?Payment \end{aligned}$$

$$\begin{aligned} PriceReq ::= {} & (\rho_C, \rho_M, \text{PRICE}, good_C \circ num_C, good_M \circ num_M, \uparrow) \\ ; {} & (\rho_C, \rho_M, \text{PRICE}, price_C, price_M, \downarrow) \end{aligned}$$

$$\begin{aligned} BuyReq ::= {} & ((price_C \geq 100?buy_C := cancelled; card_C := null; ncard_C := null) \\ \oplus {} & (price_C < 100?buy_C := accepted)) \\ ; {} & (\rho_C, \rho_M, \text{BUY}, buy_C \circ card_C \circ ncard_C, buy_M \circ card_M \circ ncard_M, \uparrow)) \end{aligned}$$

$$\begin{aligned} Order ::= {} & (\rho_M, \rho_S, \text{ORDER}, good_M \circ num_M, good_S \circ num_S, \uparrow) \\ ; {} & (\rho_M, \rho_S, \text{ORDER}, outcome_M, outcome_S, \downarrow) \end{aligned}$$

$$BuyResp ::= (\rho_M, \rho_C, \text{RESULT}, outcome_M, outcome_C, \uparrow)$$

$$\begin{aligned} Payment ::= {} & (\rho_M, \rho_B, \text{PAY}, card_M \circ ncard_M \circ price_M, card_B \circ ncard_B \circ price_B, \uparrow) \\ ; {} & (\rho_B, \rho_C, \text{RECEIPT}, receipt_B, receipt_C, \uparrow) \end{aligned}$$

Finally, we define the following initial constraints over the variables:

$$\begin{aligned} X = {} & good_C \in \{apple, banana\} \wedge \\ & \wedge\ 0 \leq num_C \leq 200 \wedge \\ & \wedge\ card_C \in \{visa, mastercard\} \wedge \\ & \wedge\ ncard_C = price_C = buy_C = receipt_C = outcome_C = \bot \wedge \end{aligned}$$

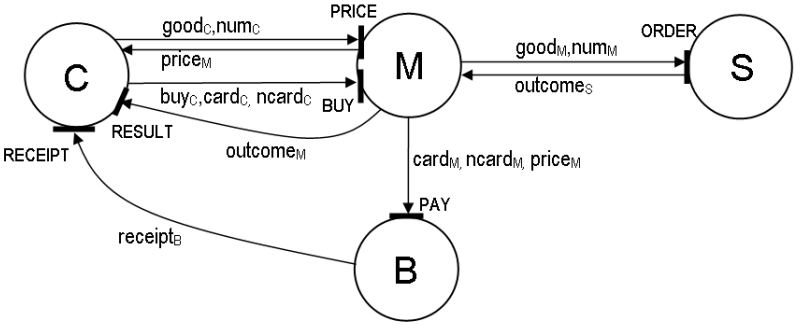

Fig. 1. Interactions among the roles

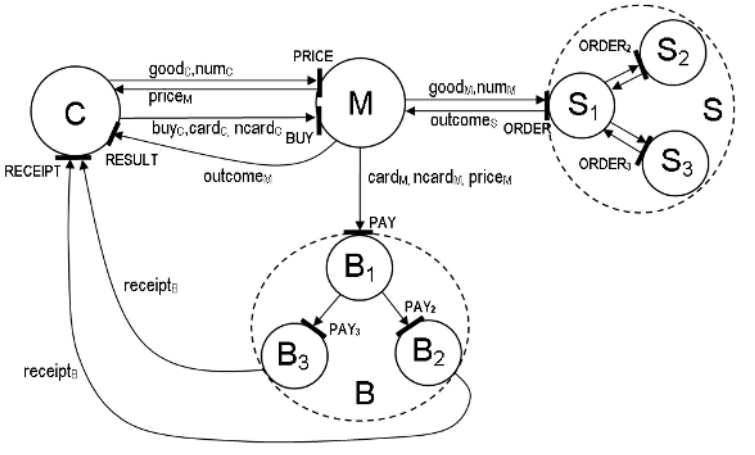

Fig. 2. Orchestration system E_2

$\wedge\ good_M = num_M = card_M = ncard_M = buy_M = outcome_M = \perp\ \wedge$
$\wedge\ 50 \leq price_M \leq 200\ \wedge$
$\wedge\ good_S = num_S = card_S = \perp\ \wedge$
$\wedge\ outcome_S \in \{OK, REJECTED\}\ \wedge$
$\wedge\ receipt_B = ReceiptDoc\ \wedge$
$\wedge\ card_B = ncard_B = price_B = \perp$

We consider the choreography $Chor = (Con, \Sigma, X)$. In Fig. 1 are graphically represented the interactions among the roles set by Con without showing the order they are performed. The circles are the roles, the bold segments are the operations and the arrows are the interactions. In the following we present two possible orchestrated systems both conformant with the choreography $Chor$. Here we intend to show that the orchestrated system can have different levels of refinement without loosing the conformance with a given choreography.

1. We consider an orchestrated system E_1 with four orchestrators: C, M, S and B whose definition follows:

$$E_1 ::= C \parallel M \parallel S \parallel B$$

$$\begin{aligned} C ::= {} & [(\overline{\text{PRICE}}(good_C \circ num_C, price_C); \\ & ; \quad ((price_C \geq 100?buy_C := cancelled; card_C := null; ncard_C := null) \\ & \oplus \quad (price_C < 100?buy_C := accepted)); \overline{\text{BUY}}(buy_C \circ card_C \circ ncard_C) \\ & ; \quad \text{RESULT}(outcome_C)) \mid \text{RECEIPT}(receipt_C)]_C \end{aligned}$$

$$\begin{aligned} M ::= {} & [\text{PRICE}(good_M \circ num_M, price_M, \mathbf{0}) \\ & \mid \quad (\text{BUY}(buy_M \circ card_M \circ ncard_M); buy_M = accepted?Ord)]_M \\ Ord ::= {} & \overline{\text{ORDER}}(good_M \circ num_M, outcome_M); (\overline{\text{RESULT}}(outcome_M) \\ & \mid \quad outcome_M = OK?\overline{\text{PAY}}(card_M \circ ncard_M \circ price_M)) \end{aligned}$$

$$S ::= [\text{ORDER}(good_S \circ num_S, outcome_S, \mathbf{0})]_S$$

$$B ::= [\text{PAY}(card_B \circ ncard_B \circ price_B); \overline{\text{RECEIPT}}(receipt_B)]_B$$

We consider a joining function Ψ where C, M, S and B embody roles ρ_C, ρ_M, ρ_S and ρ_B, respectively that is:
$\Psi^1(C) = \rho_C, \Psi^1(M) = \rho_M, \Psi^1(S) = \rho_S, \Psi^1(B) = \rho_B,$
$\Psi^1(id) = \perp \; for \; id \notin \{C, M, S, B\}.$
As far as the variables are concerned we consider a joining function projection Ψ^2 which joins the orchestrated system variables with the choreography ones that have the same name.

2. We consider a system E_2 where there are more than four orchestrators. In particular the supplier service and the bank service are splitted into three orchestrators which are joined to the same role.

$$E_2 ::= C \parallel M \parallel S_1 \parallel S_2 \parallel S_3 \parallel B_1 \parallel B_2 \parallel B_3$$

$$\begin{aligned} C ::= {} & [(\overline{\text{PRICE}}(good_C \circ num_C, price_C); \\ & ; ((price_C \geq 100?buy_C := cancelled; card_C := null; ncard_C := null) \\ & \oplus (price_C < 100?buy_C := accepted)); \overline{\text{BUY}}(buy_C \circ card_C \circ ncard_C)) \\ & ; \text{RESULT}(outcome_C)) \mid \text{RECEIPT}(receipt_C)]_C \end{aligned}$$

$$\begin{aligned} M ::= {} & [\text{PRICE}(good_M \circ num_M, price_M, \mathbf{0}) \mid \\ & \mid \quad ((\text{BUY}(buy_M \circ card_M \circ ncard_M); buy_M = accepted?Ord))]_M \\ Ord ::= {} & \overline{\text{ORDER}}(good_M \circ num_M, outcome_M); (\overline{\text{RESULT}}(outcome_M) \\ & \mid \quad outcome_M = OK?\overline{\text{PAY}}(card_M \circ ncard_M \circ price_M)) \end{aligned}$$

$$\begin{aligned} S_1 ::= {} & [\text{ORDER}(good_{S1} \circ num_{S1}, outcome_{S1}, SelS)]_{S1} \\ SelS ::= {} & good_{S1} = apple?\overline{\text{ORDER}_2}(good_{S1} \circ num_{S1}, outcome_{S1}) \\ & \oplus \; good_{S1} = banana?\overline{\text{ORDER}_3}(good_{S1} \circ num_{S1}, outcome_{S1}) \end{aligned}$$

$S_2 ::= [\mathrm{ORDER}_2(good_{S2} \circ num_{S2}, outcome_{S2}, \mathbf{0})]_{S2}$
$S_3 ::= [\mathrm{ORDER}_3(good_{S3} \circ num_{S3}, outcome_{S3}, \mathbf{0})]_{S3}$

$B_1 ::= [\mathrm{PAY}(card_{B1} \circ ncard_{B1} \circ price_{B1});$
$; (card_{B1} = visa?\overline{\mathrm{PAY}_2}(card_{B1} \circ ncard_{B1} \circ price_{B1})$
$\oplus\ card_{B1} = mastercard?\overline{\mathrm{PAY}_3}(card_{B1} \circ ncard_{B1} \circ price_{B1}))]_{B1}$
$B_2 ::= [\mathrm{PAY}_2(card_{B2} \circ ncard_{B2} \circ price_{B2}); \overline{\mathrm{RECEIPT}}(receipt_{B2})]_{B2}$
$B_3 ::= [\mathrm{PAY}_3(card_{B3} \circ ncard_{B3} \circ price_{B3}); \overline{\mathrm{RECEIPT}}(receipt_{B3})]_{B3}$

We consider the following joining function: $\Psi^1(C) = \rho_C, \Psi^1(M) = \rho_M, \Psi^1(S_1) = \rho_S, \Psi^1(S_2) = \rho_S, \Psi^1(S_3) = \rho_S, \Psi^1(B_1) = \rho_B, \Psi^1(B_2) = \rho_B, \Psi^1(B_3) = \rho_B$ $\Psi^1(id) = \perp\ for\ id \notin \{C, M, S_1, S_2, S_3, B_1, B_2, B_3\}$.

As for as the variables are concerned we exploit the same rule used for example 1 but with the following differences:

$\Psi^2(S_1)(good_{S1} = good_S)$
$\Psi^2(S_1)(num_{S1} = num_S)$
$\Psi^2(B_1)(card_{B1} = card_B)$
$\Psi^2(B_1)(price_{B1} = price_B)$
$\Psi^2(S_2)(outcome_{S2} = outcome_S)$
$\Psi^2(S_3)(outcome_{S3} = outcome_S)$
$\Psi^2(B_2)(receipt_{B2} = receipt_B)$
$\Psi^2(B_3)(receipt_{B3} = receipt_B)$

The first orchestrated system joins strongly the choreography because there is an orchestrator for each role and all the variables are the same, furthermore all the communications follow the choreography conversation. On the contrary, the second one shows how roles can be splitted on more than one orchestrator without loosing the conformance with the choreography. In particular it is worth noting that interactions within roles S and B are irrelevant to the end of conformance because they are performed between orchestrators joined with the same role. Such a kind of interaction are hidden by the $/\!/\!/ E_{id}$ operator.

6 Conclusion

In this work we continue the line of research initiated in [BGG+05] devoted to the formalization of the notion of conformance between a choreography and an orchestrated system, as well as the formalization of the notion of orchestration and choreography languages. More precisely, we extend our formal framework with the notion of state and asynchronous communication. The introduction of state is fundamental to specify the dependencies of system behavior on actual values, for instance, the fact that a customer selects one seller because it offers the best price. The second modification is useful to have a closer modeling of the way orchestrators actually communicate on, e.g., the Internet.

From a technical point of view, these extensions have required a considerable amount of work related to an appropriate modeling of nondeterminism. In particular, we had to significantly rephrase the notion of conformance. Moreover, the new notion of conformance supports the distributed implementation at the orchestration level of choreography roles. For instance, an abstract role for credit card payment can be actually implemented by means of a group of orchestrators that support the interaction between banks and credit card institutions.

The conformance notion we have defined between these two concrete languages is a powerful mechanism for designing and developing complex systems. The designer can start the design phase by programming the choreography and, in a second stage, to program and refine orchestrated systems testing, step by step, its conformance w.r.t. the choreography thus obtaining a correct implementation of the system.

As future work we intend to develop a mathematical machinery for extracting the interfaces and the workflow skeleton of the orchestrators starting from a given choreography. This will permit to verify the conformance even when the whole set of orchestrator is not completely known (the unknown orchestrators will be synthesized directly from the the choreography). As far as process calculi are concerned we intend to make a closer comparison between the two languages we propose and the most interesting proposals like WS-BPEL for orchestration and WS-CDL for choreography. In [GGL05] we present a partial comparison which investigates the interactions patterns by drawing a parallel between WS-CDL and our choreography language.

References

[BBM+05] M. Baldoni, C. Badoglio, A. Martelli, V. Patti, and C. Schifanella. Verifying the conformance of web services to global interaction protocols: a first step. In *Proc. of Web Services and Formal Methods Workshop (WS-FM'05)*, volume 3670 of *LNCS*, pages 257–271. Springer-Verlag, 2005.

[BGG+05] Nadia Busi, Roberto Gorrieri, Claudio Guidi, Roberto Lucchi, and Gianluigi Zavattaro. Choreography and orchestration: A synergic approach for system design. In *ICSOC (International Conference of Service Oriented Computing)*, pages 228–240, 2005.

[BGJ+05] T. Berg, O. Grinchtein, B. Jonsson, M. Leucker, H. Raffelt, and B. Steffen. On the Correspondence Between Conformance Testing and Regular Inference. In *Proc. of Fundmental Approaches to Software Engineering (FASE'05)*, volume 3442 of *LNCS*, pages 175–189. Springer-Verlag, 2005.

[CHYa] Marco Carbone, Kohei Honda, and Nabuko Yoshida. Programming interaction with types. [http://www.w3.org/2002/ws/chor/5/06/F2FJune14.pdf], W3C WS-CDL WG London F2F, June 14 2002.

[CHYb] Marco Carbone, Kohei Honda, and Nabuko Yoshida. A theoretical basis of communication-centred concurrent programming. Posted at w3-chor mailing list, November 2005.

[DD04] Remco Dijkman and Marlon Dumas. Service-oriented design: A multi-viewpoint approach. *Int. J. Cooperative Inf. Syst.*, 13(4):337–368, 2004.

[GGL05] R. Gorrieri, C. Guidi, and R. Lucchi. Reasoning on the interaction patterns in choreography. In *Proc. of Web Services and Formal Methods Workshop (WS-FM'05)*, volume 3670 of *LNCS*, pages 333–348. Springer-Verlag, 2005.

[HM05] R. Heckel and L. Mariani. Automatic Conformance Testing of Web Services. In *Proc. of Fundmental Approaches to Software Engineering (FASE'05)*, volume 3442 of *LNCS*, pages 34–48. Springer-Verlag, 2005.

[Kel76] Robert M. Keller. Formal verification of parallel programs. *Commun. ACM*, 19(7):371–384, 1976.

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

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

[W3C] W3C. *Web Services Choreography Description Language Version 1.0. Working draft 17 December 2004*. [http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/].