Verifying the Conformance of Web Services to Global Interaction Protocols: A First Step*

M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and and C. Schifanella

Dipartimento di Informatica — Università degli Studi di Torino,
C.so Svizzera, 185 — I-10149 Torino (Italy)
{baldoni, baroglio, mrt, patti, schi}@di.unito.it

Abstract. Global choreographies define the rules that peers should respect in their interaction, with the aim of guaranteeing interoperability. An abstract choreography can be seen as a protocol specification; it does not refer to specific peers and, especially in an open application domain, it might be necessary to retrieve a set of web services that fit in it. A crucial issue, that is raising attention, is verifying whether the business process of some peers, in particular the parts that encode the communicative behavior, will produce interactions which are conformant to the agreed protocol (legality issue). Such issue is tackled by the so called *conformance test*, which is a means for certifying the capability of interacting of the involved parts: two peers that are proved conformant to a same protocol will actually *interoperate* by producing a legal conversation. This work proposes an approach to the verification of a priori conformance of a business process to a protocol, which is based on the theory of formal languages and guarantees the interoperability of peers that are individually proved conformant.

Keywords: web service interaction protocols, conformance test, formal verification, finite state automata.

1 Introduction

In this work we propose a formal framework for verifying the conformance and the interoperability of web services with respect to a high-level specification of the global protocol. This proposal builds upon experience of protocol conformance problems in the research area of Multi-agent systems (MASs).

Web services are heterogeneous devices that can be "composed" (in a broad meaning) so as to accomplish complex tasks. Even though web services are not necessarily agents, the two share some similarities. For instance, they are usually supposed to bear an executable description of their business process, that,

* This research is partially supported by MIUR Cofin 2003 "Logic-based development and verification of multi-agent systems" national project and by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779.

M. Bravetti et al. (Eds.): EPEW 2005 and WS-FM 2005, LNCS 3670, pp. 257–271, 2005.

in particular, accounts for their interactive behavior. Similarly, agents are commonly supposed to make their communicative behavior (the agent's interaction policy) public. In both cases this description can be used to *take decisions* about the entity, such as deciding if it can take part to a system of cooperating parties.

According to Agent-Oriented Software Engineering [14], a distinction is made between the global and the individual points of view of the interaction between the various parties. The *global* viewpoint is captured by an *abstract protocol*, expressed by formalisms like AUML, automata or Petri Nets. The *local* viewpoint of one of the parties, instead, is captured by the agent's policy. Being part of the agent implementation, policies are usually written in some executable language. Having these two descriptions it is possible to decide if an agent can take a role in an interaction. In fact, this problem can be read as the problem of proving if the agent's policy *conforms* to the abstract protocol specification.

A similar need of distinguishing a global and a local view of the interaction is recently emerging also in the area of Service Oriented Architectures. In this case there is a distinction between the *choreography* of a set of peers, i.e. a global specification of the way a group of peers interact, and the concept of *behavioral interface*, seen as the specification of the interaction from the point of view of an individual peer: "The fundamental difference between the concept of choreography on the one hand, and the concept of behavioral interface (i.e., BPEL abstract process) on the other, is that a choreography focuses on interactions seen from a global viewpoint, while behavioral interfaces focus on communication actions seen from the viewpoint of one of the participants" [4]. A third concept is that of *orchestration* (e.g. BPEL executable process) which, intuitively, describes the whole service, i.e. both its communicative and its non-communicative behavior, allowing execution. The recent W3C proposal of the choreography language WS-CDL [15] is emblematic. In fact the idea behind it is to introduce specific *choreography languages* as languages for a high-level specification, captured from a global perspective, distinguishing this representation from the other two, that will be based upon ad hoc languages (like BPEL or ebXML).

Taking this perspective, choreographies and agent interaction protocols undoubtedly share a common purpose. In fact, they both aim at expressing *global interaction protocols*, i.e. rules that define the global behavior of a system of cooperating parties. The respect of these rules guarantees the interoperability of the parties (i.e. the capability of *actually* producing an interaction), and that the interactions will satisfy given requirements.

One problem that becomes crucial is the development of formal methods for verifying if the behavior of a peer respects a choreography. The applications would be various. A choreography could be used *at design time* (a priori) for verifying that the internal processes of a service enable it to participate appropriately in the interaction. At *run-time*, choreographies could be used to verify that everything is proceeding according to the agreements. A choreography could also be used unilaterally to detect exceptions (e.g. a message was expected but not received) or help a participant in sending messages in the right order and at the right time. Moreover, choreographies allow the implementation of a top-down

methodology in the design of web services. The work in [7] already takes this approach by using WS-CDL and BPEL4WS as complementary design tools: the first design step of an interaction protocol (for the peers of an e-commerce system) consists in the development of a WS-CDL description; this is followed by an implementation step, where BPEL4WS is exploited for representing the behavior of the single peers [7]. A further step could be exploiting formal methods for synthesizing behavioral interfaces (e.g. abstract BPEL) from the choreography definition, on the line of the work in [8].

In the literature the problem of verifying conformance of the behavior of an individual to a general interaction protocol is known as *conformance* testing. A conformance test can be considered as a tool that, by verifying the respect of a protocol, *certifies* the interoperability of a set of parties: we expect that two parties which are proved conformant to a same protocol *will produce* an interaction, that is legal w.r.t. the encoded rules, when they will interact. In the last years two kinds of conformance have been studied w.r.t. MASs [12]: *a priori* conformance (checked at design time) [9,10], and *run-time* conformance [3,1]. If we call a *conversation* a specific interaction between two agents, consisting only of communicative acts, the former is a property of the *implementation as a whole* –intuitively it checks if an agent will never produce conversations that violate the abstract interaction protocol specification–, while the latter is a property of the *on-going conversation*, aimed at verifying if *that* conversation is legal. Notice that the *same tests* are envisioned for choreographies and for the individual peers that should play a role defined in them.

In this work we focus on testing *a priori conformance* and develop a framework based on the use of formal languages. In our framework a global interaction protocol (a choreography), is represented as a finite state automaton, whose alphabet is the set of messages exchanged among peers. It specifies permitted conversations. Atomic services (peers), that have to be composed according to the choreography, are described as finite state automata as well. Given such a representation we capture a concept of conformance that answers positively to all these questions: *is it possible to verify that a peer, playing a role in a given global protocol, produces at least those conversations which guarantee interoperability with other conformant peers? Will such a peer always follow one of these conversations when interacting with the other parties in the context of the protocol? Will it always be able to conclude the legal conversations it is involved in?* Technically, the conformance test is based on the acceptance of both the peer's behavior and the global protocol by a special finite state automaton. The interesting characteristic of this test is that it guarantees the interoperability of peers that are proved conformant *individually* and *independently* from one another.

This approach can be applied to a wide variety of cases with the proviso that both the protocol specification and the behavioral interface can be specified by regular expressions. Besides simplicity and readability, the reason for adopting regular expressions is that they guarantee *decidability*. Of course, in this way it is not possible to represent *concurrency*. We are aware of this limit but this is just a first step of a wider research, and we mean to extend the approach in

the near future. Focussing on finite state automata is not too much restrictive, anyway, because many protocols used in MASs can be expressed in this way and we believe that the same holds for many web services. So far, the framework only deals with 2-party global protocols. This is also, of course, a limitation that we aim to relax in future work by extending the framework (see the conclusions for further discussion). To make this proposal more concrete in Section 4 we explain these ideas with the help of an example, in which we consider a choreography and a web service; we show that the latter conforms to the former and, thus, it will be able to interoperate with any other service that is as well conformant and that plays another role.

2 Conformant and Interoperable Peers

A business process is a program that defines the behavior of a specific peer, implemented in some programming language. We focus on the interactive behavior of the peer and we will denote it by the term *conversation policy* of the peer. A choreography specifies the overall behavior of a group of interacting peers; many proposals of languages (e.g. WSCI and WS-CDL) for representing choreographies can be found in the literature. Also in this case we will focus only on that part of the choreography that denotes the message exchange among the parties. For this reason, hereafter the term *choreography* and the term (conversation) *protocol* will be used as synonims.

We face the problem of conformance verification by interpreting "a priori conformance" as a property that relates two *formal languages*: the language of the conversations allowed by the conversation policy of a peer, and the language of the conversations allowed by a choreography. They will respectively be denoted by $L(p^{ws}_{lang})$ and $L(p_{spec})$, where *spec* is the choreography specification language, *lang* is the language in which the policy, executed by the peer ws, is written, and p is the name of the policy or of the protocol at issue. The assumption that we do throughout this paper is that the two languages are *regular sets*. This choice restricts the kinds of protocols to which our proposal can be applied, because finite state automata cannot represent concurrent operations, however, it is still significant because a wide family of protocols (and policies) of practical use can be expressed in a way that can be mapped onto such automata. Moreover, the use of regular sets ensures decidability. Another assumption is that the conversation protocol encompasses only *two peers*. The extension to a greater number of peers will be tackled as future work. Notice that the peers might be implemented in *different languages*.

A conversation protocol specifies the sequences of messages that can possibly be exchanged by the involved peers, and that we consider as legal. In agent languages that account for communication, messages (named "speech acts") often have the form $m(ag_s, ag_r, l)$, where m is the performative, ag_s (sender) and ag_r (receiver) are two agents and l is the message content. It is not restrictive to assume that messages have this form also in the case of web services and to assume that conversations are sequences of messages of this form [2]. In the

following analysis it is important to distinguish the incoming messages, w.r.t. a specific peer ws, from the messages sent by it. We respectively denote the former, where ws plays the role of the receiver, by $\mathsf{m}(\overleftarrow{ws})$, and the latter, where ws is the sender, by and $\mathsf{m}(\overrightarrow{ws})$. We will also simply write $\overleftarrow{\mathsf{m}}$ (*incoming message*) and $\overrightarrow{\mathsf{m}}$ (*outgoing message*) when the peer that receives or sends the message is clear from the context. Notice that these are just short notations, that underline the *role* of a given peer from the *individual perspective* of *that* peer. This view is consistent with the unilateral view typical of languages like BPEL [6], used to represent behavioral interfaces from the point of view of a peer. So, for instance, $m(ws_s, ws_r, l)$ is written as $\mathsf{m}(\overleftarrow{ws_r})$ from the point of view of ws_r, and $\mathsf{m}(\overrightarrow{ws_s})$ from the point of view of the sender but the three notions denote the same object.

A *conversation*, denoted by σ, is a sequence of messages that represents a dialogue of a set of peers. We say that a conversation is legal w.r.t. a protocol if it respects the specifications given by the protocol. Since $L(p_{spec})$ is the set of all the legal conversations according to p, the definition is as follows.

Definition 1 (Legal conversation). *We say that a conversation σ is legal w.r.t. a protocol specification p_{spec} when $\sigma \in L(p_{spec})$.*

We can now explain, with the help of simple examples, the intuition behind the terms "conformance" and "interoperability", that we will then formalize.

> *Interoperability is the capability of a peer of actually producing a conversation when interacting with another.*

A new peer can be introduced in an execution context provided that it satisfies the rules of the system. As long as this happens, it will not be necessary to verify interoperability with the single components of the system. This can be done by checking the interactive behavior of the peer against the rules of the group, i.e. against its *interaction protocol*. Such a proof is known as *conformance test* and must, intuitively, guarantee the following expectations.

> *We expect that two peers, that conform to a protocol, will produce a legal conversation, when interacting with one another.*

Let us begin with considering the following case: suppose that the communicative behavior of the peer ws is defined by a policy that accounts for two conversations $\{\mathsf{m}_1(\overrightarrow{ws})\mathsf{m}_2(\overleftarrow{ws}), \mathsf{m}_1(\overrightarrow{ws})\mathsf{m}_3(\overleftarrow{ws})\}$. This means that after sending a message m_1, the peer expects one of the two messages m_2 or m_3. Let us also suppose that the protocol specification only allows the first conversation, i.e. that the only possible incoming message is m_2. Is the policy conformant? According to Def. 1 the answer should be no, because the policy allows an illegal conversation. Nevertheless, when the peer will interact with another peer that is conformant to the protocol, the message m_3 will never be received because the partner will never send it. So, in this case, we would like the a priori conformance test to accept the policy as *conformant* to the specification.

Talking about incoming messages, let us now consider the symmetric case, in which the *protocol specification* states that after the peer ws has sent m_1, the

other peer can alternatively answer m_2 or m_4 (ws's policy, instead, is the same as above). In this case, the expectation is that ws's policy is *not conformant* because, according to the protocol, there is a possible legal conversation (the one with answer m_4) that can be enacted by the *interlocutor* (which is not under the control of ws), which ws cannot handle. So it does not comply to the specifications.

> *As a first observation we expect the policy to be able to handle any incoming message, foreseen by the protocol, and we ignore those cases in which the policy foresees an incoming message that is not supposed to be received at that point of the conversation, according to the protocol specification.*

Let us, now, suppose that peer ws's policy can produce the following conversations $\{\mathsf{m}_1(\overleftarrow{ws})\mathsf{m}_2(\overrightarrow{ws}),\ \mathsf{m}_1(\overleftarrow{ws})\mathsf{m}_3(\overrightarrow{ws})\}$ and that the set of conversations allowed by the protocol specification is $\{\mathsf{m}_1(\overleftarrow{ws})\mathsf{m}_2(\overrightarrow{ws})\}$. Trivially, this policy is *not conformant* to the protocol because ws can send a message (m_3) that cannot be handled by any interlocutor that is conformant to the protocol.

> *The second observation is that we expect a policy to never send a message that, according to the specification, is not supposed to be sent at that point of the conversation.*

Instead, in the symmetric case in which the policy contains only the conversation $\{\mathsf{m}_1(\overleftarrow{ws})\mathsf{m}_2(\overrightarrow{ws})\}$ while the protocol states that ws can answer to m_1 alternatively by sending m_2 or m_3, *conformance holds*. The reason is that at any point of its conversations the peer will always send legal messages. The restriction of the set of possible alternatives (w.r.t. the protocol) depends on the peer implementor's own criteria. However, the peer must foresee *at least one* of such alternatives otherwise the conversation will be interrupted. Trivially, the case in which the policy contains only the conversation $\{\mathsf{m}_1(\overleftarrow{ws})\}$ is *not conformant*.

> *The third observation is that we expect that a policy always allows the peer to send one of the messages foreseen by the protocol at every point of the possible conversations. However, it is not necessary that a policy envisions all the possible alternatives.*

To summarize, at every point of a conversation, we expect that a conformant policy never sends messages that are not expected, according to the protocol, and we also expect it to be able to handle any message that can possibly be received, once again according to the protocol. However, the policy is not obliged to foresee (at every point of conversation) an outgoing message for every alternative included in the protocol (but it must foresee at least one of them). Incoming and outgoing messages are, therefore, *not handled in the same way*.

These expectations are motivated by the desire to define a minimal set of conditions which assure the construction of a conformance test that guarantees the *interoperability* of peers. We claim –and we will show– that two peers that respect this minimal set of conditions (w.r.t. an agreed protocol) will *actually*

be able to interact, respecting the protocol. The relevant point is that this certification is *a property that can be checked on each single single peer, rather than on the choreographed system as a whole.*

3 Conformance Test

In order to decide if a policy is conformant to a protocol specification, it is not sufficient to perform an inclusion test; instead, as we have intuitively shown by means of the above examples, it is necessary to prove mutual properties of both $L(p^{ws}_{lang})$ and $L(p_{spec})$. The method that we propose, for proving such properties, consists in verifying that both languages are recognized by a special finite state automaton, whose construction we are now going to explain. Such an automaton is based on the automaton that accepts the *intersection* of the two languages. This, however, is not sufficient, because there are further conditions to consider, for instance there are conversations that we mean to allow but that do not belong to the intersection.

3.1 The Automaton M_{conf}

If $L(p^{ws}_{lang})$ and $L(p_{spec})$ are regular, they are accepted by two (deterministic) *finite automata*, that we respectively denote by $M(p^{ws}_{lang})$ and $M(p_{spec})$, that we can assume as having the *same alphabet* (see [13]). An automaton is a five-tuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite input alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and δ is a transition function mapping $Q \times \Sigma$ to Q. In a finite automaton we can always classify states in two categories: *alive states*, that lie on a path from the initial state to a final state, and *dead states*, the other ones. Intuitively, alive states accept the language of the prefixes of the strings accepted by the automaton.

For reasons that will be made clear shortly, we request the two automata to show the following property: the edges that lead to a same state must *all* be labelled either by incoming messages or by outgoing messages w.r.t. ws.

Definition 2 (IO-automaton). *Given an automaton $M = (Q, \Sigma, \delta, q_0, F)$, let $E_q = \{m \mid \delta(p, m) = q\}$ for $q \in Q$. We say that M is an IO-automaton iff for every $q \in Q$, E_q alternatively consists only of incoming or only of outgoing messages w.r.t. a peer ws.*

Notice that an automaton that does not show this property can always be transformed so as to satisfy it, in linear time w.r.t. the number of states, by splitting those states that do not satisfy the property. We will denote a state q that is reached only by incoming messages by the notation $\overleftarrow{q}$ (we will call it an I-state), and a state q that is reached only by outgoing messages by $\overrightarrow{q}$ (an O-state).

Finally, let us denote by $M^{\times}(p^{ws}_{lang}, p_{spec})$ the deterministic finite automaton that accepts the language $L(p^{ws}_{lang}) \cap L(p_{spec})$. It is defined as follows. Let $M(p^{ws}_{lang})$ be the automaton $(Q^P, \Sigma, \delta^P, q^P_0, F^P)$ and $M(p_{spec})$ the automaton $(Q^S, \Sigma, \delta^S, q^S_0, F^S)$:

$$M^{\times}(p^{ws}_{lang}, p_{spec}) = (Q^P \times Q^S, \Sigma, \delta, [q_0^P, q_0^S], F^P \times F^S)$$

where for all q^P in Q^P, q^S in Q^S, and m in Σ, $\delta([q^P, q^S], m) = [\delta^P(q^P, m), \delta^S(q^S, m)]$. We will briefly denote this automaton by $M^{\times}$.

Notice that all the *conversations* that are accepted by $M^{\times}$ are surely *conformant* (Def. 1). For the so built automaton, it is easy to prove the following property.

Proposition 1. *$M^{\times}(p^{ws}_{lang}, p_{spec})$ is an IO-automaton if $M(p^{ws}_{lang})$ and $M(p_{spec})$ are two IO-automata.*

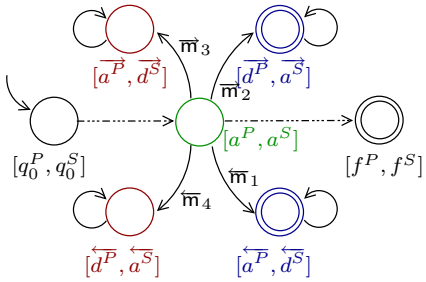

Fig. 1. A general schema of the M_{conf} automaton. From bottom-right clockwise cases (a), (b), (c), and (d).

Definition 3 (Automaton M_{conf}). *The finite state automaton $M_{conf}(p^{ag}_{lang}, p_{spec})$ is built by applying the following steps to $M^{\times}(p^{ag}_{lang}, p_{spec})$ until none is applicable:*

(a) if $\overleftarrow{q} = [\overleftarrow{a^P}, \overleftarrow{d^S}]$ in Q is an I-state, such that $\overleftarrow{a^P}$ is an alive state and $\overleftarrow{d^S}$ is a dead state, we set $\delta(\overleftarrow{q}, m) = \overleftarrow{q}$ for every m in Σ, and we put $\overleftarrow{q}$ in F;

(b) if $\overleftarrow{q} = [\overleftarrow{d^P}, \overleftarrow{a^S}]$ in Q is an I-state, such that $\overleftarrow{d^P}$ is dead and $\overleftarrow{a^S}$ is alive, we set $\delta(\overleftarrow{q}, m) = \overleftarrow{q}$ for every m in Σ, without modifying F;

(c) if $\overrightarrow{q} = [\overrightarrow{a^P}, \overrightarrow{d^S}]$ in Q is an O-state, such that $\overrightarrow{a^P}$ is alive and $\overrightarrow{d^S}$ is dead, we set $\delta(\overrightarrow{q}, m) = \overrightarrow{q}$ for every m in Σ (without modifying F);

(d) if $\overrightarrow{q} = [\overrightarrow{d^P}, \overrightarrow{a^S}]$ in Q is an O-state, such that $\overrightarrow{d^P}$ is dead and $\overrightarrow{a^S}$ is alive, we set $\delta(\overrightarrow{q}, m) = \overrightarrow{q}$ for every m in Σ, and we put $\overrightarrow{q}$ in F.

These four transformation rules can, intuitively, be explained as follows. Rule (a) handles the case in which, at a certain point of the conversation, according to the policy it is possible to receive a message that, instead, cannot be received according to the specification (it is the case of message $\overleftarrow{\mathsf{m}}_1$ in Figure 1). Actually, if the peer will interact with another peer that respects the protocol, this message can never be received, so we can ignore the paths generated by the policy from the message at issue onwards. Since this case does not compromise

conformance, we want our automaton to accept all these strings. For this reason we set the state as final. Rule (b) handles the symmetric case (Figure 1, message $\overleftarrow{\mathsf{m}}_4$), in which at a certain point of the conversation it is possible, according to the specification, to receive a message, that is not accounted for by the implementation. In this case the state at issue is turned into a trap state (a state that is not final and that has no transition to a different state); by doing so, all the conversations that are foreseen by the specification from that point onwards will not be accepted by M_{conf}. Rule (c) handles the cases in which a message can possibly be sent by the peer, according to the policy, but it is not possible according to the specification (Figure 1, message $\overrightarrow{\mathsf{m}}_3$). In this case, the policy is not conformant, so we transform the current state in a trap state. By doing so, part of the conversations possibly generated by the policy will not be accepted by the automaton. The symmetric case (Figure 1, message $\overrightarrow{\mathsf{m}}_2$), instead, does not prevent conformance, in fact, a peer is free not to send a message foreseen by the protocol. However, the conversations that can be generated from that point, according to the latter, are to be accepted as well. For this reason the state is turned into an accepting looping state.

One may wonder if the application of rules (b) and (c) could prevent the *reachability of states*, that have been set as accepting states by the other two rules. Notice that their application cannot prevent the reachability of *alive-alive* accepting states, i.e. those that accept the strings belonging to the intersection of the two languages, because all the four rules only work on dead states. If a state has been set as a trap state (either by rule (b) or (c)), whatever conversation is possibly generated after it by the policy is illegal w.r.t. the specification. So it is correct that the automaton is modified in such a way that the policy language is not accepted by it and that the final state cannot be reached any more.

3.2 Conformance and Interoperability

We can now discuss how to check that a peer conforms to a given protocol. The following is a first definition of conformance, that guarantees the expectations that we have explained by examples in Section 2. That is: the peer will always send, at any point of conversation, messages that are legal according to p_{spec} (though it is not necessary that it foresees all the alternatives), and it will be able to handle at least every incoming message, expected by the protocol. A first attempt of defining conformance is the following.

Definition 4. *A policy p^{ws}_{lang} is conformant to a protocol specification p_{spec} iff the automaton $M_{conf}(p^{ws}_{lang}, p_{spec})$ accepts both languages $L(p^{ws}_{lang})$ and $L(p_{spec})$.*

The following proposition underlines the *role of the public protocol* of representing the set of *all the possible interlocutors.*

Proposition 2. *All the conversations that a policy p^{ws}_{lang}, that is conformant according to Def. 4 to a protocol specification p_{spec}, will produce when it interacts with any peer that is equally conformant to p_{spec}, are always legal w.r.t. this protocol, according to Def. 1.*

Proof. Let us consider the general schema of M_{conf} in Figure 1. If p^{ws}_{lang} is conformant, $L(p^{ws}_{lang})$ is accepted by M_{conf}. Then, by construction M_{conf} does not contain any state $[\overrightarrow{a^P}, \overrightarrow{d^S}]$ due to illegal messages sent by the peer nor it contains any state $[\overleftarrow{d^P}, \overleftarrow{a^S}]$ due to incoming messages that are not accounted for by the policy. Obviously, no conversation σ in $L(p^{ws}_{lang})$ can be accepted by states of the kind $[\overrightarrow{d^P}, \overrightarrow{a^S}]$ because the peer does not send the messages required to reach such states. Finally, no conversation produced by the send will be accepted by states of the kind $[\overleftarrow{a^P}, \overleftarrow{d^S}]$ if the interlocutor is also conformant to the protocol, because the latter cannot send illegal messages. **q.e.d.**

In other words, whatever conversation is in the intersection $\cap^{i=1,2}_{ws_i} L(p^{ws_i}_{lang_i})$, where $p^{ws_i}_{lang_i}$, $i = 1, 2$ are the conversation policies of two peers that conform to p_{spec}, it is legal. However, we would like conformance to have a *stronger implication*: if two peers, playing the two roles of a same protocol, are proved conformant to it, we would like *each of them* to be able to lead to an end all the conversations *it is involved in* by the other peer (which will respect the protocol). Def. 4 guarantees the satisfaction of the first two expectations reported in Section 2, however, it is *not enough to guarantee* the above statement (third expectation), which requires that, at every state of the conversation, if a role is supposed to send a message out of a set of possibilities, the peer's policy envisions *at least one* of them.

Given $L(p_{spec})$ and $L(p^{ws}_{lang})$, let us consider $M(p_{spec}) = (Q^S, \Sigma, \delta^S, q^S_0, F^S)$ and $M_{conf}(p^{ws}_{lang}, p_{spec}) = (Q^P \times Q^S, \Sigma, \delta, [q^P_0, q^S_0], F_{conf})$. Let us consider those states $q^S \in Q^S$, that emit edges labelled with outgoing messages, w.r.t. ws, which are part of strings accepted by $M(p_{spec})$ (legal conversations according to the protocol specification). More formally, for each such state q^S there is at least one $\mathsf{m}(\overrightarrow{ws})$ such that $\delta^S(q^S, \mathsf{m}(\overrightarrow{ws})) = p^S$ and p^S is an *alive state*. We will denote by $Mess_{q^S}$ the set of all such messages.

Definition 5 (Complete automaton). *We say that the automaton M_{conf} is* complete *iff for all states of form $[q^P, q^S]$ of M_{conf}, such that $Mess_{q^S} \neq \emptyset$, there is a message $\mathsf{m}(\overrightarrow{ws})' \in Mess_{q^S}$ such that $\delta([q^P, q^S], \mathsf{m}(\overrightarrow{ws})')$ is a state of M_{conf} composed of two alive states.*

Definition 6 (Policy conformance test). *A policy p^{ws}_{lang} is conformant to a protocol specification p_{spec} iff the automaton $M_{conf}(p^{ws}_{lang}, p_{spec})$ is complete and it accepts both languages $L(p^{ws}_{lang})$ and $L(p_{spec})$.*

We are now in condition to state that a policy that passes the above test can carry on *any* conformant conversation it is involved in.

Theorem 1. *Given a policy p^{ws}_{lang} that is conformant to a protocol specification p_{spec}, according to the test in Def. 6, for every prefix σ' that is common to the two languages $L(p_{spec})$ and $L(p^{ws}_{lang})$, there is a conversation $\sigma = \sigma'\sigma''$ such that σ is in the intersection of $L(p^{ws}_{lang})$ and $L(p_{spec})$.*

Proof. (*sketch*) If σ' is a common prefix, then it leads to a state of the automaton M_{conf} of the kind $[a^P, a^S]$ (see Figure 1). By the same reasons on which the proof of Prop. 2 is based, *if* there is a conversation $\sigma = \sigma'\sigma''$ in $L(p^{ws}_{lang})$, then this must be a legal conversation. Now, at every step after the state $[a^P, a^S]$ mentioned above, due to *policy conformance* all the incoming messages (w.r.t. the peer) must be foreseen by the policy. Moreover, due to the *completeness* of M_{conf}, in the case of outgoing messages, the policy must foresee at least one of them. Therefore, from $[a^P, a^S]$ it is possible to perform one more common step. **q.e.d.**

Notice that the *intersection* of $L(p^{ws}_{lang})$ and $L(p_{spec})$ cannot be empty because of policy conformance, and also that Theorem 1 does not entail that the two languages coincide (i.e. the policy is not necessarily a full implementation of the protocol). As a consequence, given that the conversation policies of two peers ws_1 and ws_2, playing the different roles of an interaction protocol p_{spec}, are *conformant* to the protocol, according to Def. 6, and denoting by I the intersection $\cap^{i=1,2}_{ws_i} L(p^{ws_i}_{lang_i})$, we can prove ws_1 and ws_2 *interoperability.* The demonstration is similar to the previous one. Roughly, it is immediate to prove that every prefix that is common to the two policies also belongs to the protocol, then, by a reasoning process that is close to the previous demonstration, it is possible to prove that a common legal conversation must exist

Proposition 3 (Interoperability). *For every prefix σ' that is common to the two languages $L(p^{ws_1}_{lang_1})$ and $L(p^{ws_2}_{lang_2})$, there is a conversation $\sigma = \sigma'\sigma''$ such that $\sigma \in I$.*

Starting from regular languages, all the steps that we have described that lead to the construction of M_{conf} and allow the verification of policy conformance, are decidable and the following theorem holds.

Theorem 2. *Policy conformance is decidable when $L(p^{ws}_{lang})$ and $L(p_{spec})$ are regular languages.*

4 An Example

Let us, now, show by means of an example how the proposed conformance test works. Given an interaction protocol (a choreography) and the interaction policy of a specific web service, we mean to verify (a priori, at design time) if the web service fits the interaction schema encoded by the choreography, from the point of view of one of the roles (the role that the peer should play). We will, then, verify its a priori conformance. Given that conformance holds, we are guaranteed that the service will be able to interoperate with any other service, equally proved conformant to the protocol, that will play the other foreseen role. The protocol is reported in a graphical notation in Fig. 2. It is very simple: the peer that plays the role "cinema" waits for a request from another peer (the request is whether a certain movie is played); then, it can alternatively send the requested information (yes or no) or refuse to supply information; the protocol is ended by an acknowledgement from the customer to the cinema.

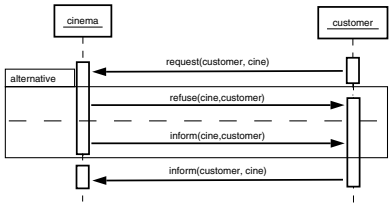

Fig. 2. The interaction protocol as an AUML sequence diagram [17]

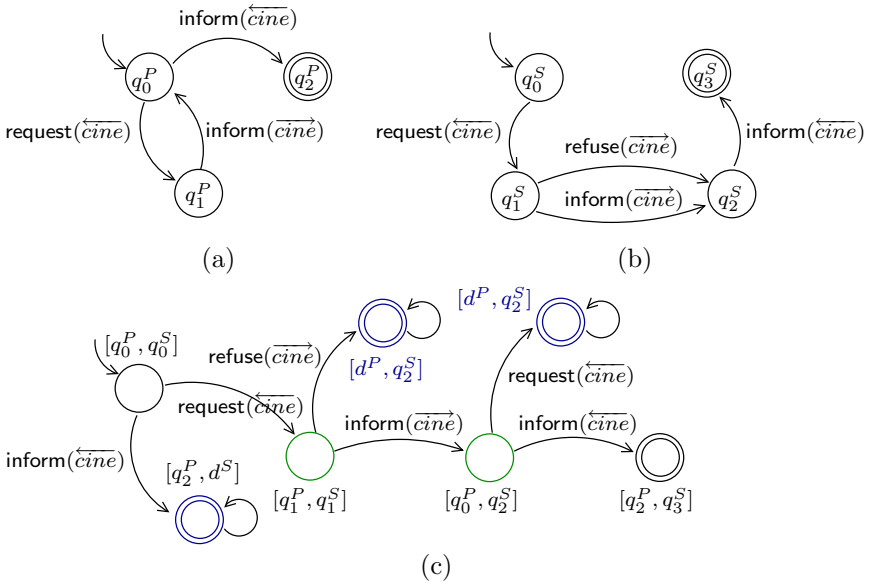

Fig. 3. (a) Policy of agent *cine*; (b) global protocol specification; (c) M_{conf} automaton. Only the part relevant to the discussion is shown.

The peer's policy could, for instance, be described in an executable business process language, such as BPEL4WS. Actually, in the literature other authors have already proposed algorithms for extracting a formal representation from a BPEL representation. For instance, Viroli [18] proposes a formal semantics for this language, focussing right on the message exchange and correlation sets. It is not difficult to see that, disregarding the operator that concerns concurrency (*flow*), the exception and fault handlers, and correlation sets, it is possible to turn a BPEL description in a regular language (i.e. a finite state automaton). Fig. 3 (a) reports a finite state automaton that represents the interactive behavior of our cinema service, [1] Briefly, the web service has a *reactive behavior* and it is not trivial to see that it conforms to the protocol: it waits for a message; if it is

[1] The program of the customer is not given: we will suppose that it adheres to the public and global choreography, against which we check the peer's conformance.

a request from a customer, then it (always) supplies the requested information; if it is an acknowledgement it stops. In the remainder of the paper we will refer to this web service by the name *cine*. For what concerns the choreography, we can say something similar, at least for what concerns the current proposal for WS-CDL. If we ignore the constructs for dealing with concurrency it is possible to turn a choreography in an automaton. The automaton reported in Fig. 3(b), for instance, is obtained straightforwardly from the WS-CDL representation reported in the Appendix.

The question that we want to answer is whether *cine*'s policy is *conformant* to the given protocol, and we will discuss whether another agent that plays as a customer and that is proved conformant to the protocol will actually be able to *interoperate* with this particular player of the cinema role. [2] The protocol allows only two conversations between *cine* and *customer* (the content of the message is not relevant in this example, so we skip it): request(*customer*, *cine*) inform(*cine*, *customer*) inform(*customer*, *cine*) and request(*customer*, *cine*) refuse(*cine*, *customer*) inform(*customer*, *cine*). Let us denote this protocol by $\mathsf{get\_info\_movie}_{WSCDL}$ (WS-CDL is the specification language). Let us now consider the web service *cine*. The service's behavior depends on the message that it receives, and its policy allows an infinite number of conversations of any length. Let us denote this language by $\mathsf{get\_info\_movie}^{cine}_{BPEL}$ (BPEL should be the implementation language). In general, it allows all the conversations that begin with a (possibly empty) series of exchanges of kind $\mathsf{request}(\overleftarrow{cine})$ followed by $\mathsf{inform}(\overrightarrow{cine})$, concluded by a message of kind $\mathsf{inform}(\overleftarrow{cine})$.

To verify its conformance to the protocol, and then state its interoperability with other peers that respect such protocol, we need to build the M_{conf} automaton for its policy and the protocol specification. For brevity, we skip its construction steps and directly report M_{conf} in Fig. 3(c). Let us now analyze M_{conf} for answering our queries. Trivially, the automaton is *complete* and it *accepts both languages* (of the policy and of the protocol), therefore, $\mathsf{get\_info\_movie}^{cine}_{BPEL}$ is policy conformant to $\mathsf{get\_info\_movie}_{WSCDL}$. Moreover, when the service interacts with another service *customer* whose policy is conformant to $\mathsf{get\_info\_movie}_{WSCDL}$, the messages $\mathsf{request}(\overleftarrow{cine})$ and $\mathsf{inform}(\overleftarrow{cine})$ will not be received by *cine* in all the possible states it expects them. The reason is simple: for receiving them it is necessary that the interlocutor utters them, but by definition (it is conformant) it will not. The fact that $\mathsf{refuse}(\overrightarrow{cine})$ is never uttered by *cine* does not compromise conformance and interoperability.

5 Conclusions and Future Work

In this work we propose a formal framework that can be applied for verifying the conformance and the interoperability of web services with respect to a global protocol definition which is meant to be provided at the choreography level. The idea is that a choreography definition can be exploited *at design time*

[2] Notice that in Fig. 3 all the short notations for the messages are to be interpreted as incoming or outgoing messages w.r.t. the cinema service.

for verifying that the internal processes of a web service will enable it to participate appropriately in the choreography. For achieving this goal we need a formal framework for specifying both general interaction protocols and web services' local interaction policies. We proposed a framework based on the theory of *formal languages*, where both the global protocol and the web service behavior are expressed by using finite state automata. Finite-state automata have been adopted also by Berardi et al. [5] but for web service composition.

Within this framework we formalize a notion of *a priori* conformance (see Def. 6), having some important property. First, it guarantees that the service, at any point of its conversations, can only send messages which are legal w.r.t. the global interaction protocol, because of the M_{conf} construction step, given by rule (c). Moreover it guarantees that the service will be able to handle any incoming message, foreseen by the protocol. Notice that the service may also expect incoming messages, that are not expected by the protocol specification, for this does not prevent the correct interaction with another conformant service. Finally, it guarantees that the service will always send at least one of the messages foreseen by the protocol, although it is not necessary that its policy envisions all the possible alternatives (e.g. the designer can restrict the set of the possible answers). All these properties define a minimal set of conditions which, on the one hand, ensure the preservation of the *interoperability* of the peers, while, on the other hand, they give some flexibility in designing service policies.

As we explained from the very beginning the current choice of finite state automata bears some serious limitation: the impossibility of tackling concurrency. Moreover, the framework so far can only check conformance of a service, whose behavioral interface contains interactions with only another service. Multi-party interaction is not tackled. These limitations were, in a way, necessary to allow the identification of a set of concepts and of conditions that characterize interoperability and its verification: the first step of the work. As future directions of research, however, it is mandatory to study, on the one hand, the possible extensions to policies that encode the interaction with many parties, and on the other to study whether it is possible to decide conformance in presence of concurrency, by adopting more expressive kinds of automata. For what concerns the first problem we think (but still have to prove) that the test as it is now could quite easily be extended so as to tackle unilateral interactions with many parties which do not interact with one another. For what concerns the latter problem, instead, it will be necessary to identify alternative representations. For instance, process algebras are formal tools that are commonly used for verifying properties of interacting processes, we could study whether and how to apply them to prove a property like conformance. Also concurrent regular expressions [11] should be investigated. Last but not least, a crucial point is the semantics of the languages used for representing choreographies and behavioral interfaces (or orchestrations), e.g. BPEL4WS and WS-CDL, which is not precisely defined yet. The absence of a formal semantics is, indeed, an obstacle to the automation of property check in service oriented applications. Concerning BPEL4WS, some proposal of a formal semantics exists and the proposed formal methods derive

from formal models for concurrency and coordination of distributed systems (e.g. process algebras) [18,16].

References

1. M. Alberti, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Specification and verification of agent interactions using social integrity constraints. In *Proc. of the Workshop on Logic and Communication in Multi-Agent Systems, LCMAS 2003*, volume 85(2), Eindhoven, the Netherlands, 2003. Elsevier.
2. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services*. Springer, 2004.
3. M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella. Verifying protocol conformance for logic-based communicating agents. In *Proc. of CLIMA V*, LNCS series 2005. To appear.
4. A. Barros, M. Dumas, and P. Oaks. A critical overview of the web services choreography description language(ws-cdl). *Business Process Trends*, 2005. http://www.bptrends.com.
5. D. Berardi, D. Calvanese, G. G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In *Proc. of ICSOC 2003*, *LNCS* 2910, pages 43–58. Springer, 2003.
6. BPEL4WS. http://www-106.ibm.com/developerworks/library/ws-bpel. 2003.
7. M. Bravetti, C. Guidi, R. Lucchi, and G. Zavattaro. Supporting e.commerce systems formalization with choreography languages. In *Proc. of SAC'05*. ACM Press, 2005.
8. T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: A new approach to design and analysis of e-service composition. In *Proc. of WWW'03*, 2003.
9. U. Endriss, N. Maudet, F. Sadri, and F. Toni. Protocol conformance for logic-based agents. In *Proc. of IJCAI-2003*, pages 679–684. 2003.
10. U. Endriss, N. Maudet, F. Sadri, and F. Toni. Logic-based agent communication protocols. In *Advances in agent communication languages*, *LNAI* 2922, pages 91–107. Springer-Verlag, 2004.
11. V. Garg and M.T. Ragunath. Concurrent regular expressions and their relationship to Petri nets. *Theoretical Computer Science*, 96:285–304, 1992.
12. F. Guerin and J. Pitt. Verification and Compliance Testing. In *Communication in Multiagent Systems*, *LNAI* 2650, pages 98–112. Springer, 2003.
13. J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Company, 1979.
14. M. P. Huget and J.L. Koning. Interaction Protocol Engineering. In *Communication in Multiagent Systems*, *LNAI* 2650, pages 179–193. Springer, 2003.
15. N. Kavantzas, D. Burdett, G. Ritzinger, and Y. Lafon. Web services choreography description language version 1.0. Available at http://www.w3.org/TR/ws-cdl-10, 2004.
16. M. Mazzara and R. Lucchi. A framework for generic error handling in business processes. In *Proc. of WS-FM 2004*, volume 105 of *ENTCS*. Elsevier, 2004.
17. J. Odell, H. V. D. Parunak, and B. Bauer. Extending UML for agents. In *Proc. of the Agent-Oriented Information System Workshop at AAAI'00*. 2000.
18. M. Viroli. Towards a formal foundation to orchestration languages. In *Proc. of WS-FM 2004*, volume 105 of *ENTCS*, pages 51–71, Eindhoven, the Netherlands, 2004. Elsevier.