Choice, Interoperability, and Conformance in Interaction Protocols and Service Choreographies

Matteo Baldoni
Dipartimento di Informatica
Università degli Studi di Torino
C.so Svizzera, 185
I-10149 Torino, Italy
baldoni@di.unito.it

Cristina Baroglio
Dipartimento di Informatica
Università degli Studi di Torino
C.so Svizzera, 185
I-10149 Torino, Italy
baroglio@di.unito.it

Amit K. Chopra
Dept. of Computer Science
North Carolina State Univ.
Raleigh, NC 27695-8206, USA
akchopra.mail@gmail.com

Nirmit Desai
IBM India Research Labs
Embassy Golf Links, Block D
Bangalore 560071, India
nirmitv@gmail.com

Viviana Patti
Dipartimento di Informatica
Università degli Studi di Torino
C.so Svizzera, 185
I-10149 Torino, Italy
patti@di.unito.it

Munindar P. Singh
Dept. of Computer Science
North Carolina State Univ.
Raleigh, NC 27695-8206, USA
singh@ncsu.edu

ABSTRACT

Many real-world applications of multiagent systems require independently designed (heterogeneous) and operated (autonomous) agents to interoperate. We consider agents who offer *business services* and collaborate in interesting business service engagements. We formalize notions of *interoperability* and *conformance*, which appropriately support agent heterogeneity and autonomy. With respect to autonomy, our approach considers the choices that each agent has, and how their choices are coordinated so that at any time one agent *leads* and its counterpart *follows*, but with initiative fluidly shifting among the participants. With respect to heterogeneity, we characterize the variations in the agents' designs, and show how an agent may conform to a specification or substitute for another agent. Our approach addresses a challenging problem with *multiparty* interactions that existing approaches cannot solve. Further, we introduce a set of *edit* operations by which to modify an agent design so as to ensure its conformance with others.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; D.2.4 [**Software/Program Verification**]: Formal methods; H.3.5 [**Online Information Services**]: Web-based services

General Terms

Design, Verification, Theory

Keywords

Interaction Protocols, Interoperability, Conformance, Choreographies, Web Services

1. INTRODUCTION

Cite as: Choice, Interoperability, and Conformance in Interaction Protocols and Service Choreographies , M. Baldoni, C. Baroglio, A. K. Chopra, N. Desai, V. Patti, M. P. Singh, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

The accomplishment of a complex task often requires interactions among a set of parties. For instance, in a business process scenario, a seller may need to interact with a payment service and a shipper in order to support a purchase. These partners must coordinate their executions and must be able to interact with each other. There is broad agreement on the importance of describing such interactions formally. The agents community refers to such a specification as an *interaction protocol*, whereas the services community refers to it as a *choreography*. In deference to the services literature and because we do not study higher-level notions such as commitments, we use the term *choreography* in this paper. A choreography abstractly specifies an interaction in terms of its *roles*, the messages exchanged among the roles, and constraints on the message exchanges. An agent playing a role in a choreography would send and receive some subset of the messages the role is specified to send and receive.

Agents who offer business services and collaborate with each other in business service engagements must be *interoperable* [5, 8, 13, 20, 27]. In open environments, such as the Web, where coalitions are formed dynamically, and the partners can be *replaced* or *upgraded* at any moment, their interoperability is guaranteed by adopting a choreography whose roles are by design interoperable. Proving that the actors of the various roles *conform* to the corresponding specification ensures their interoperability. When a business partner (for instance, a shipping service) becomes unavailable, we would need replace it with a new partner that we are confident will support the interactions that the other partners had with the original service. Similar considerations apply when one of the parties (e.g., the payment service) is upgraded, maybe to allow a richer interaction. This approach, which relies on the notion of *conformance* [5, 6, 9, 10, 19, 24, 25], is highly practical: (1) it enables the distribution of the interoperability test, and (2) it does not require *any knowledge* about the other services involved in the interaction. The only assumption is that all of them respect the agreed upon choreography. Conformance guarantees substitutability: namely, that *substituting a role (or an old implementation) by a conformant player preserves the interoperability of the service composition*.

The social ability of agents to interact with others should, moreover, be reconciled with the possibility of *interpreting* the choreography according to own goals and further policies, rather than to implement it strictly [22]. For instance, to realize a choreography,

a set of business services must be discovered or implemented that play the various roles but no existing service may perfectly match a given role. Further, a designer may need to customize a service implementation, for instance, to take into account specific privacy norms. Still the policy that is adopted must be acceptable. For this reason, we aim at capturing the *degree of freedom* that is affordable when modifying the specified behavior, so to remain *conformant* and, then, interoperable w.r.t. the choreography. In particular, we set up a framework in which any amount of design variation is possible as long as conformance is preserved. The key question is *how to transform a conformant implementation into another conformant implementation*. We show that the proposed framework supports a set of *edit operations* that allows such transformations. Based on this, it will be possible to evaluate, before the interaction takes place, to which extent a partner is free in deciding about its behavior, having a means for practically producing modifications of it. So edit operations can be used to produce *upgrades* of existing parties as well as to correct in some cases their flows of interaction, producing appropriate *patches* (or adaptors), so to make them interoperable with interlocutors they could not interact with before.

A common characteristic of approaches to conformance is a distinction of the way incoming messages (*receptions*) and outgoing messages (*emissions*) are handled: usually, the assumption is that in every state of the interaction a service will either be a sender or a receiver, and that the initiative about which message to exchange is up to the sender. Let us consider, for example, a book-selling service that offers operations such as *search* and *buy* to its customers. When interacting with a customer, the service will await the customer's request and will not initiate a search or the selling procedure. The book-selling service's execution will depend on a message sent by the customer. On the other hand, once the book-selling service arrives at a state of execution when it must give some information to its customer, for instance, that a book is not available or that it is possible to proceed with the order, the choice of which message to send depends on the book-selling service, i.e., on its *internal* computation.

The above yields a simple definition of conformance. Assuming the seller interoperates with the customer, any service that entertains all the incoming messages that the seller entertains and produces no more messages than the seller produces will also interoperate with the customer. That is, the new service is conformant with the seller. Bordeaux *et al.* [8] codify this intuition as the motto *less emissions, more receptions!* Approaches to conformance, such as [8, 5, 6, 9, 10, 19, 24, 25], analyse the messages that are exchanged at every step and allow substituting an existing participant with a new participant that may not exactly match the specification but produces narrower sets of emissions and tackles broader sets of receptions, without compromising interoperability.

However, making the assumption that the initiative of choosing the action to perform necessarily lies with the sender and that, conversely, each party must be able to tackle all possible receptions, is limitating and does not account for communication models where interaction is performed, for instance, by publishing and reading information. Consider, as an example, a *surveillance system* where a Monitor interacts with a set of Sensors. Each Sensor makes a continuously updated temporal series of data available to the Monitor through a *blackboard system*. The Monitor, on the other hand, decides which data to read at every step, depending on its internal policies. In this case, the information producer, the Sensor, does not choose which information to provide: it is up to the consumer to decide, the Sensor must have the whole series ready at any time. Even more general is the case where the Monitor can either decide to read some data on the blackboard or to send a message to the Sensor and ask it to change some parameter of its configuration.

This example shows that the receipient of information, in our case the Monitor, can exercise control on what it receives. In general, in *cross-organizational business management* [16] we need to model interactions very flexibly: in particular, at any step an autonomous participant can either decide to say something or to wait for a message [13]. Accordingly, we treat making or accepting a choice as *orthogonal* to sending and receiving a message. Thus, at any decision point, a service may (as leader) make a choice to send or to receive or (as follower) accept its partner's choice to send or to receive.

This paper makes the following contributions. First, it proposes and formalizes notions of interoperability and conformance that are centered on an explicit representation of *decision points*, i.e., of those points of the interaction in which a party *chooses*, and those in which it *accepts* its partner's choices. Second, it establishes key theoretical results involving these definitions, especially regarding the preservation of interoperability through the substitution of a conforming agent for another. Third, it expands this treatment to accommodate multiparty interactions, which otherwise can lead to deadlocks, and which previous approaches cannot handle. Fourth, it defines a set of *edit operations* that allow the construction of conformant variants by: (a) *reducing* or *augmenting* the alternatives an agent has at some decision point, depending on whether the agent has the *lead* on that specific choice or it has to *follow* the partner's decision; (b) *splitting* or *merging* decision points, once again depending on whether it has the lead or not.

The rest of this paper is as follows. Section 2 expands on the intuition behind our proposal. Section 3 develops the notions of interoperability and conformance suited for cross-organizational business interactions. Section 4 presents a set of edit operations that can be applied to the problems of service update and patch production. Section 5 concludes with a discussion of the relevant literature.

2. PROTOCOLS, CHOICES, AUTONOMY

In any system made of interacting parties, decision points and the rules by which one or more of the parties are entitled to take the initiative play a crucial role. Specifically, agents are usually considered autonomous. Yet, fully autonomous agents would have difficulty interoperating. In fact, since they have the power to take their own decisions independently from one another, they need to *negotiate* at every step what to do next. So, if we consider two agents offering the business services s_1 and s_2, where (say, at a particular stage in their interaction) s_1 can send messages m_1, m_2, and m_3, while s_2 can receive m_2, m_3, and m_4, they will have to agree on exchanging either m_2 or m_3 in order to interoperate. Negotiation is a costly and time-consuming process. A simplification is to assume that, at every step, one of the two services *is explicitly entitled to decide* and may autonomously choose how to act, thus taking the initiative in the interaction, while the other must be capable of tackling whatever decision the first service takes.

Deciding which action to perform is nontrivial for the agent that holds the initiative because some choices can compromise the continuation of the interaction. Following the above example, suppose that s_1 has the initiative of choosing. Now, if s_1 decides to send m_1, that will compromise s_1's interaction with s_2. Similarly, the interaction can be compromised if the choice is up to s_2 (the receiver) and it decides to read m_4. So s_1 and s_2 should agree that only m_2 and m_3 can be successfully exchanged. More practically, to this aim it is possible to use protocols. Interaction protocols and choreographies encode how the initiative is distributed among the parties and the messages that can be exchanged. Conformance tests, e.g. those in [8, 5, 6, 9, 10, 19, 24, 25] allow substituting an

existing participant with a new participant that produces narrower sets of emissions and tackles broader sets of receptions, without compromising interoperability. However, in the Survaillance System example, the Monitor *leads* by exercising choice on which data to read, and the Sensor *follows* by sending a corresponding message. Notice that if the Monitor wishes to pick data that is not supported by the Sensor, the interaction will fail.

The approach, that we propose, defines protocols so as the initiative is naturally shared among the parties, by making *leads* and *follows* explicit and by describing parties in such a way to flexibly hand over the initiative to one another.

To support such generality, we treat the initiative of choosing as *orthogonal* to whether a service is sending or receiving a message. Consequently, we base our definitions of interoperability and conformance on the notion of "initiative to choose" itself rather than on the kind of action (sending or receiving) on which the choice applies, and we adopt a new motto:

lead less, follow more!

As we show, the notions of "leading" (i.e. being entitled of making a choice) and "following" (i.e. delegating the choice to a partner) come in handy for computing the degree of freedom that is available in modifying an implementation.

The approach that we propose is abstract and sits well with different kinds of infrastructure; specifically, both with a message exchange infrastructure and with a blackboard infrastructure. In more general terms, the traditional approaches are focused on the infrastructure level. For example, if sending a message amounts to a master invoking a method on a slave, we can imagine that the receiver be ready to execute all methods that are included in its public interface. Or even if we think of placing a request message on messaging middleware, it still is the choice of the sender. Such interactions could possibly be achieved by lower-level coordination messages in the infrastructure, but it is most valuable to focus the study of interoperability and conformance on business communications rather than on implementation details of the underlying infrastructure.

3. CONFORMANCE & INTEROPERABILITY

Let us introduce a simple representation of the behavior of the interacting parties. In the spirit of interaction, our concern is with messages sent and received, and the choices that underlie such message exchanges. As in [27, 8], we suppose that the communication model is *synchronous*. Messages have the general form $\mathsf{m}(s, r, l)$ where m is the kind of message, s and r are the sender and the receiver, and l is the content. When the receiver, sender, and content are clear from the context or are not relevant, we simply denote a message by its kind m. As in CCS [23], we use the notation $!\mathsf{m}$ to represent an outgoing message (a message that is uttered or an *emission*) and $?\mathsf{m}$ to represent an incoming message (a message that is expected or a *reception*). Moreover, given that m denotes an interactive action (either sending or receiving a message), $\overline{\mathsf{m}}$ denotes its complementary action.

Choreography roles and their players are represented in the same way, as individual processes that exchange messages with others. Whenever it is not necessary to distinguish roles from players, we use the general term "parties" to identify the entities that are involved in a definition or a result. Along the lines of [5, 7, 8, 11], the semantics of a party behavior is given in terms of *automata theory*, annotating when necessary each state with a label, to express the kind of branching structure it represents, whether a lead, $\odot$, or a follow, $\oplus$. So, for instance, a $\odot$ state with two outgoing edges, that are respectively labeled by m_1 and m_2, denotes the choice between the two interactive actions m_1 and m_2. Notice that such actions can either correspond to sending or receiving messages, and that mixed cases are also allowed. Conversely, a $\oplus$ state with two outgoing edges, labeled by m_1 and m_2 respectively, denotes the fact that the service is ready to execute both actions and will follow the choice of its partner. Also in this case there is no constraint on which kind of message (send or receive) is to be used with this operator. This generalization makes our proposal more abstract than others in the literature, suiting not only message-exchange communication infrastructures (typical of service-oriented applications) but also blackboard-based communication infrastructures. Here, in fact, it is typical to have agents that offer alternative items, and it is up to the taker to select an option.

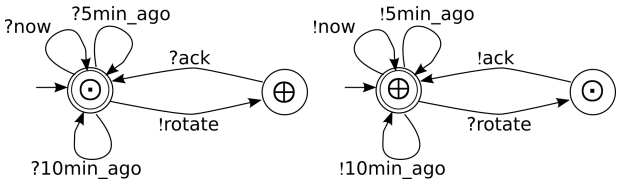

Figure 1: The survaillance interaction protocol: left the Monitor role, right the Sensor role.

DEFINITION 1 (FINITE STATE AUTOMATON). *A finite state automaton is a tuple* $(S, s_0, \gamma, \Sigma, T, F)$*, where* S *is a finite set of states,* $s_0 \in S$ *is a distinguished initial state,* γ *is function from* S *to the set* $\{\odot, \oplus, \varepsilon\}$*,* Σ *is the alphabet,* $T \subseteq (S \times \Sigma \times S)$ *is a set of transitions,* $F \in S$ *is a set of final states.*

For simplicity, we call the states labeled by $\odot$, $\oplus$ as $\odot$-states, $\oplus$-states respectively. Σ contains a set of either receptions $?\mathsf{m}$ or emissions $!\mathsf{m}$. Final states are those corresponding to the possible conclusions of the interaction. 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.

In this work we do not consider the translation process necessary to turn a choreography into a set of FSAs; our focus is, in fact, conformance and interoperability. It is possible to find in the literature some works that do this kind of translations for WS-CDL and BPEL; an example can be found in [17].

As an example of FSA, let us consider the Monitor role of the *Survaillance* protocol, represented as in Fig. 1: the Monitor takes the initiative and iteratively reads data supplied by the Sensor (either the most recent data or the one taken five or ten minutes ago). Alternatively, it can also ask the Sensor to perform an action (rotate). When this happens the Monitor waits for an acknowledgement by its partner before restarting its readings. Notice that the Sensor must continuously supply all the data its partner can read among, e.g. by writing it on the blackboard.

3.1 Interoperability

Intuitively, a set of parties is interoperable when it is *stuck-free*, i.e., whatever point of interaction may be reached, communication will not be blocked, and each of the parties will reach one of its final states [25, 5]. In other words, if we focus on a state of the execution of a set of parties, either each party has reached a final state (and the system has as well), or we expect some communication to occur between two parties. This communication will transition the system to another state of its execution. In this context, we call a *transition of the system* a successful communication, i.e., the send-

ing of a message !m, done by one of the involved parties, joint with its reception ?m, performed by another party of the system.

DEFINITION 2 (TRANSITIONS). *Let $P_1, P_2, \ldots, P_n$ be n parties, we denote by* transitions$(\langle P_1.s_{i_1}, P_2.s_{i_2}, \ldots, P_n.s_{i_n}\rangle)$ *the set* $\{\langle P_1.s_{i'_1}, P_2.s_{i'_2}, \ldots, P_n.s_{i'_n}\rangle \mid \exists h, k, m,\ 1 \leq h \neq k \leq n,\ (P_h.s_{i_h},\ m,\ P_h.s_{i'_h}) \in P_h.T$ *and* $(P_k.s_{i_k},\ \overline{m},\ P_k.s_{i'_k}) \in P_k.T,$ *and* $\forall j \neq h$ *nor* $k, P_j.s'_{i_j} = P_j.s_{i_j}\}$.

A sequence of transitions makes a *run*; in turn, a run is a *successful communication* when it takes all parties to go from their initial state to one of their final states.

DEFINITION 3 (RUNS). *Let $P_1, P_2, \ldots, P_n$ be n parties, a sequence of transitions* $(\langle P_1.s_k, P_2.s_k, \ldots, P_n.s_k\rangle, m_1, (\langle P_1.s_{k+1}, P_2.s_{k+1}, \ldots, P_n.s_{k+1}\rangle), \ldots, (\langle P_1.s_{k+l-1}, P_2.s_{k+l-1}, \ldots, P_n.s_{k+l-1}\rangle), m_l, (\langle P_1.s_{k+l}, P_2.s_{k+l}, \ldots, P_n.s_{k+l}\rangle)$, *such that, for all* $h, 1 \leq h < m, \langle P_1.s_{k+h}, P_2.s_{k+h}, \ldots, P_n.s_{k+h}\rangle$ *belongs to* transitions$(\langle P_1.s_{k+h-1}, P_2.s_{k+h-1}, \ldots, P_n.s_{k+h-1}\rangle)$.

DEFINITION 4 (SUCCESSFUL COMMUNICATION). *Let $P_1, P_2, \ldots, P_n$ be n parties, a* successful communication *is a run σ that starts from $\langle P_1.s_0, P_2.s_0, \ldots, P_n.s_0\rangle$ and ends with $\langle P_1.s_t, P_2.s_t, \ldots, P_n.s_t\rangle$, where $P_1.s_t, P_2.s_t, \ldots, P_n.s_t$ are final states.*

What are the conditions that guarantee a successful communication? The simple intuition is that at every point of the conversation, the party that leads the choice cannot pursue an alternative that cannot be tackled by its interlocutor. Conversely, the party that follows the choice must be able to handle all of the alternatives its interlocutor can choose. When this happens we say that the two parties are in *compatible* states. Let s be a state of an automaton A, we define as $message(s)$ as the set $\{m \mid \exists s', (s, m, s') \in A.T\}$. More formally, compatibility is defined as follows.

DEFINITION 5 (COMPATIBILITY). *Let P and P' be two parties and let $P.s_i \in P.S^{\odot}$ and $P'.s_j \in P'.S^{\oplus}$ be two states. We say that $P.s_i$ is* compatible *with $P'.s_j$ if*

$$message(P.s_i) \subseteq message(P'.s_j)$$

Interoperability verifies the compatibility of two parties by checking it for every pair of states that can be reached by the defined transitions.

DEFINITION 6 (INTEROPERABILITY). *Let $P_1, P_2, \ldots, P_n$ be n parties. $P_1, P_2, \ldots, P_n$ are interoperable iff there exists a n-ary relation $\mathcal{R}$ such that:*

1. $P_1.s_0\ \mathcal{R}\ P_2.s_0\ \mathcal{R}\ \ldots\ \mathcal{R}\ P_n.s_0$;
2. *if* $P_1.s_{i_1}\ \mathcal{R}\ P_2.s_{i_2}\ \mathcal{R}\ \ldots\ \mathcal{R}\ P_n.s_{i_n}$, *then:*
   - *there are h and k, $1 \leq h \neq k \leq n$, such that $P_h.s_{i_h}$ is compatible with $P_k.s_{i_k}$;*
   - *for all* $\langle P_1.s_{i'_1}, P_2.s_{i'_2}, \ldots, P_n.s_{i'_n}\rangle$ *in* transitions$(\langle P_1.s_{i_1}, P_2.s_{i_2}, \ldots, P_n.s_{i_n}\rangle)$, *we have that* $P_1.s_{i'_1}\ \mathcal{R}\ P_2.s_{i'_2}\ \mathcal{R}\ \ldots\ \mathcal{R}\ P_n.s_{i'_n}$;
   - $P_1.s_{i'_1}, P_2.s_{i'_2}, \ldots, P_n.s_{i'_n}$ *are alive.*

This notion of interoperability applies *a priori*, i.e., it checks that whatever interaction is started by the partners, they will be able to carry it to an end, each arriving at one of its final states. In particular, at every step there must be a party, whose current state is an $\odot$-state, and another party, that is in a compatible $\oplus$-state. The next proposition immediately follows from the above definition, by reasoning by absurd.

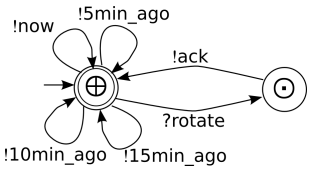

Figure 2: The Survaillance protocol: a modified Sensor, which also supplies data collected fifteen minutes ago.

PROPOSITION 3.1. *Let $P_1, P_2, \ldots, P_n$ be n interoperable parties. Let $\langle P_1.s_{i_1}, P_2.s_{i_2}, \ldots, P_n.s_{i_n}\rangle$ be a set of reachable states after a certain run σ, then there is a run σ' such that $\sigma\sigma'$ is a successful communication.*

3.2 Conformance

Now that we have defined a notion of interoperability, we can define a notion of conformance that preserves interoperability after substitution. Since any of the messages that the leader can select must be handled by the follower, reducing the set of choices preserves interoperability. Conversely, for the same reason, it is not possible to reduce the set of messages that the follower is expected to handle, although it is possible to augment this set. So, for instance, a follower that can tackle the incoming messages $?yes$, $?oui$, and $?si$ will always use only the first two alternatives when dealing with a chooser that can only say $!yes$ and $!oui$. In terms of web services, this amounts to having implementations that can accomplish a set of operations and can deal with interlocutors who always request a subset of such operations. Last but not the least, it is not possible to augment the set of messages of the leader because doing so can cause deadlocks. For example, a party that, in some state, can choose among $!yes$, $!oui$, or $!si$ cannot safely substitute a party that can choose only between $!yes$ and $!oui$. Specifically, if the original leader was able to interact with another, capable of tackling both $?yes$ and $?oui$, the new chooser can start an interaction by sending $!si$, which unfortunately cannot be tackled by the follower. As another example, consider a variant, reported in Figure 2, of the Sensor role of the Survaillance protocol and how it interacts with the Monitor role. Here the Sensor can additionally supply to its partner data that were recorded fifteen minutes ago (*15min_ago*). It is easy to see that such modifications do not compromise the interoperability with any service that is conformant to the regular Monitor role, because the choice is up to the Monitor: the Sensor can supply additional data but only if requested. For this reason there will never be any dangling communication.

The definition of *conformance* strictly depends on the notion of interoperability and, therefore, it *must preserve* the selected notion of *compatibility*. To this end, we need to define a notion of *state alignment*, which relates two states, one belonging to a party representation, the other belonging to another party representation that we would like to substitute for the former. Alignment, as well as compatibility, are properties of states.

DEFINITION 7 (ALIGNMENT). *Let P and P' be two parties. We say that the state $P'.s_j$ aligns with $P.s_i$ if:*

1. $P.s_i \in P.S^{\odot}$, $P'.s_j \in P'.S^{\odot}$, *then* $message(P.s_i) \supseteq message(P'.s_j)$ *(i.e.,* ***lead less!****);*
2. $P.s_i \in P.S^{\oplus}$, $P'.s_j \in P'.S^{\oplus}$, *then* $message(P.s_i) \subseteq message(P'.s_j)$. *Moreover, all messages have in common the same leader* [1] *(i.e.,* ***follow more!****).*

[1]The need of having a same leader will be discussed in Section 3.3.

This notion derives directly from Definition 5. It is easy to prove the following property:

PROPOSITION 3.2. *Let P_1 and P_2 be two parties, such that the state $P_1.s_i$ is compatible with $P_2.s_j$. Let P_1' and P_2' be two other parties, such that $P_1'.s_i'$ aligns to $P_1.s_i$ and $P_2'.s_j'$ aligns to $P_2.s_j$. Then, $P_1'.s_i'$ is compatible with $P_2'.s_j'$.*

We now give a formal definition of *conformance*. This notion is inspired by (bi)-simulation, which supports the comparison of processes with *different branching structures*. The present definition generalizes the proposal in [5] and is characterized by a distinguished way in which the messages of the leader and follower are handled. Interoperability verifies the alignment of states for the whole automaton. The capability of comparing parties with different branching structures adds flexibility and enables distinguishing cases in which the advancement or postponement of decision points compromises interoperability from cases in which it does not compromise interoperability. The definition that we introduce below naturally applies when interactions are ruled by choreography, which specifies various roles. Thus a likely scenario is that we find a role specification from a choreography and then locate a service that would conform that role. In this perspective, P is a role specification in a given choreography, while P' is a finite state automaton representing a players. Nevertheless, this notion can be used more generally because both P and P' can be player implementations. The consequences of this generalization will be clear in Section 4.

DEFINITION 8 (CONFORMANCE). *Let P and P' be two parties. We say that P' conforms to P iff there is binary relation $\mathcal{R}$ such that:*

1. *$P.s_0 \ \mathcal{R} \ P'.s_0$;*
2. *if $P.s_i \ \mathcal{R} \ P'.s_j$, then:*
   - *$P'.s_j$ aligns to $P.s_i$;*
   - *$\forall P'.s_{j+1}$ such that $(P'.s_j, \mathsf{m}, P'.s_{j+1})$, there is $P.s_{i+1}$ such that $(P.s_i, \mathsf{m}, P.s_{i+1})$ and $P.s_{i+1} \ \mathcal{R} \ P'.s_{j+1}$;*
3. *for all states $P'.s_j$ that are final states for P', and for which there is a state $P.s_i$, such that $P.s_i \ \mathcal{R} \ P'.s_j$, $P.s_i$ is final;*
4. *for all states $P.s_i$ that are alive and for which there is a state $P'.s_j$ such that $P.s_i \ \mathcal{R} \ P'.s_j$, $P'.s_j$ is alive.*

Intuitively, condition 2 captures the ***lead less, follow more*** intuition; condition 3 guarantees that the refined party P' (e.g., a given implementation) cannot have final states that are not foreseen by the party P; condition 4 ensures that executions of the refined party P' with a prefix that is foreseen by service P must progress toward a final state. Notice that this notion of conformance allows cutting out some executions but does not allow cutting them all away. In fact, when reducing a set of possible messages it is not possible to cut all of the alternatives.

Conformance and *interoperability* are related by the following fundamental theorem that can be proved by absurd from Definition 8, Proposition 3.2, and Proposition 3.1.

THEOREM 3.1 (SUBSTITUTABILITY). *Let $P_1, P_2, \ldots, P_n$ be a set of interoperable parties and let $P_1', P_2', \ldots, P_n'$ another set of parties such that P_i' conforms to P_i, $i = 1, \ldots, n$. Then $P_1', P_2', \ldots, P_n'$ are interoperable.*

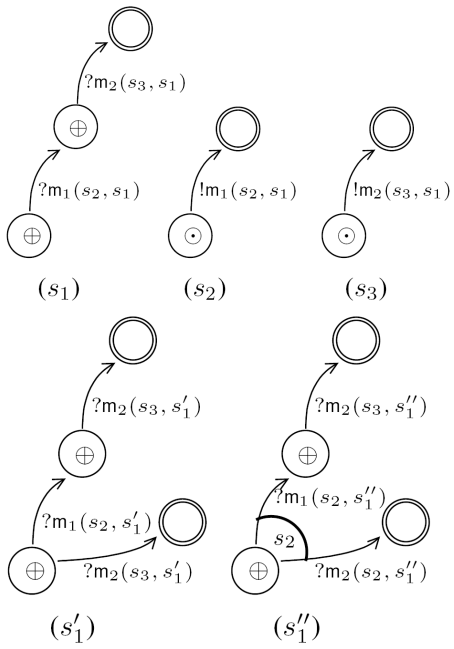

Figure 3: Substituting s_1 by s_1' causes a race condition, while substituting it with s_1'' does not; the arc in s_1'' expresses the fact that both alternatives are expected from s_2.

3.3 One Lead for One Follow

Multiparty interactions complicate the situation for choice followers. Fig. 3 illustrates an example involving three parties [25]. Here, s_1 first follows s_2 to receive message m_1, and then follows s_3 to receive m_2. The system $\{s_1, s_2, s_3\}$ is interoperable. Let s_1' be a party (Fig. 3) that is like s_1 but can handle more requests than s_1. For brevity, let s_2' and s_3' be identical to s_2 and s_3 except that they communicate with s_1'. The system $\{s_1', s_2', s_3'\}$ is not interoperable, because the message $\mathsf{m}_2(s_3, s_1')$ might be consumed before it would have been in the original system and this could produce a deadlock. Our approach yields a natural solution. The above problem arises because we have a *race condition* between partners s_2 and s_3. This race condition can compromise the interaction. In other words, s_1' will follow any of two potential leaders, s_2 and s_3. The specification includes no interaction to allow these parties to reach an agreement on who is entitled to take the lead. In fact, each of s_2 and s_3 has no option but to send its message, leaving the choice to s_1', but s_1' as the follower cannot choose. By interpreting this example in our framework, it is clear that the problem (this contradiciton) is due to an *underspecification* of who will be the leader, thus causing a race condition. Consequently, we determine an interaction such as the one expressed by s_1' to be invalid.

In order to avoid race conditions, the definition of alignment of states, which is the basic building block of our notion of conformance, includes a simple test for *follow* states: the test enforces the fact that all the possible alternatives depend on the same leader. So going back to the above example, we find that s_1' is not conformant to s_1. Further, we find a party (s_1'' in Fig. 3) to be conformant to s_1, because in its initial state, s_1'' explicitly specifies that it follows s_2, and thus remains perfectly interoperable with s_2 and s_3. Rajamani and Rehof [25] cannot satisfactorily address this problem. They simply require that the number of receptions *not* be expanded, so they would find s_1'' to be nonconformant with s_1.

The restriction to a single leader per follow state may seem a

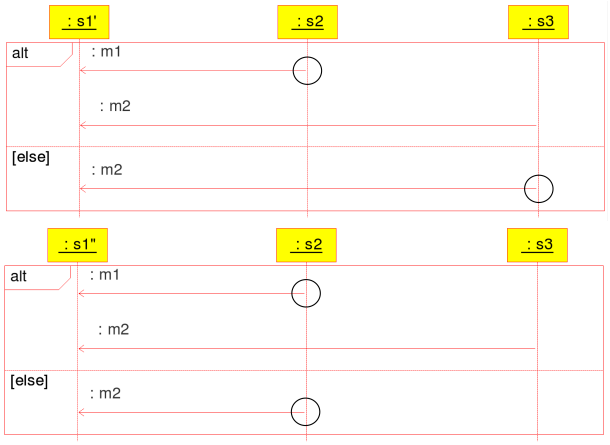

Figure 4: UML sequence diagram for s_1', s_2, and s_3 (on the top: circles underline the race condition between s_2 and s_3, both of them may send a message at the same time) and for s_1'', s_2, and s_3 (on the bottom: there is no race condition).

little strong, however, if we consider the corresponding *UML sequence diagram* (see Fig. 4, top) the alternative lies along the lifeline of s_1'; *it is not possible to define the alternative in other ways because the other two parties have separate lifelines and have no option*. So, according to the UML definition of *alternative*, [1, p. 454], the only party to have a choice of behavior is s_1' (the choice between receiving a message from s_2 or receiving one from s_3). The only way for s_1' to delegate the choice is that the alternatives lie on the same lifeline of one of its interlocutors, i.e. that the leader is unique (Fig. 4, bottom).

4. COMPATIBLE UPDATES AND PATCHES

This section introduces a set of derivation rules for producing conformant refinements that preserve interoperability. We introduce a set of *edit operations* denoted by the symbol $\Rightarrow$; by $S \Rightarrow^* S'$ we represent the fact that by applying a sequence of edit operations to the party S, we obtain the party S'. The name "edit operations" is taken from the literature on versioning systems and automatic document change detection. In a choreography, S will be a role specification and the rules will produce conformant implementations of S. Indeed, scripts made of these rules capture the differences between a party and a variant of it.

DEFINITION 9 (DERIVABILITY RELATION). *The derivability relation $\Rightarrow^*$ between two parties is defined as the transitive, reflexive, and contextual closure of the relation $\Rightarrow$, defined by the unidirectional transformations in Fig. 5.*

The *Follow More* rule captures the possibility of expanding choices where a party would follow another. We suppose that the choices that are added correspond to messages that do not already appear as choices in the party on the left-hand side, otherwise conformance could be compromised. The *Lead Less* rule captures the possibility of reducing choices where a party leads. The other rules correspond to *changing the branching structure* of a party. When merging follow choices the expected interlocutor for all the merged states must be the same. The above edit operations are a means to enable the derivation of a conformant implementation either from a role specification or from a player that is known to be conformant to such a specification. In both cases, the transformations guarantee the new party's interoperability (Theorem 4.1). Thus, any of the derivable

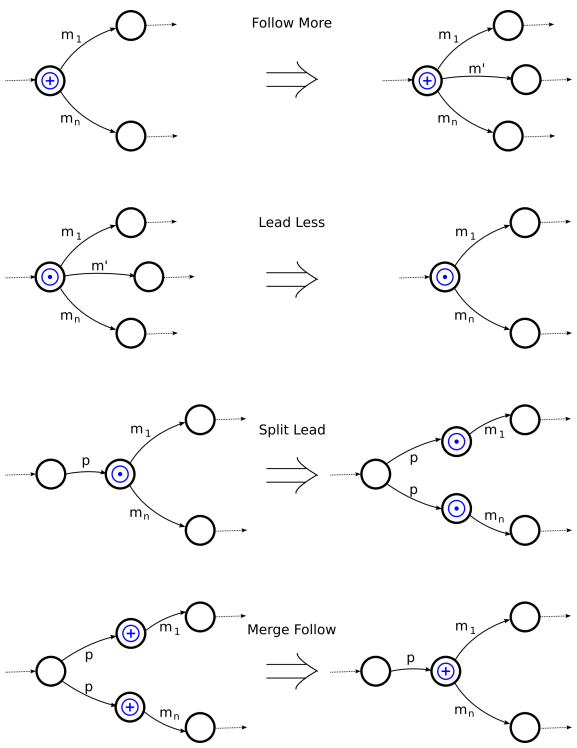

Figure 5: Edit operations.

variants can be chosen *safely*.

THEOREM 4.1. *Let S and S' be two parties. If S' is derivable from S, i.e., $S \Rightarrow^* S'$, then S' conforms to S.*

The derivation relation, in essence, specifies a designer's freedom in specializing the party behavior, while keeping it safely adherent to the reference specification.

DEFINITION 10 (SPACE OF CANDIDATES). *Let S be a* role specification. *Then, the set of players that are conformant to S, written as* candidates(S), *is given by:*

$$\textsf{candidates}(S) = \{I \mid I \text{ is a player } \wedge (S \Rightarrow^* I)\}$$

In some application contexts, however, the appropriate role specification may be only indirectly available. For instance, a specification may not be disclosed for business reasons. In such a case, the possible variants of the specification S (i.e., candidates(S)) would not be known. However, an existing interoperable party would still be available. The above results enable us to use an existing party as a reference implementation with respect to which we can produce possible variants. This is useful in a *party update* scenario, where a party is substituted by a new version and we want the upgraded one to be usable by clients designed for the old version. Therefore, a player S can be substituted by a player S' (obtained by applying an edit path), being sure that backward compatibility is guaranteed. The transformation rules drive the upgrade process, by defining the lawful changes. In this case, the previous definition can be turned into the following:

DEFINITION 11 (CONFORMANT PARTY UPDATES). *Let I be a* party. *Then* update(I) *is defined as* candidates(I).

Conformance need not be checked again. It is easy to prove that if a service s_0 conforms to a specification s_1 which, in turn, is interoperable with a role s_2, then s_0 will be interoperable with *any* conformant implementation s_3 of s_2. Since s_0' belongs to update(s_0),

s_0' conforms to s_0 and, therefore, also to s_1; thus, it is interoperable with s_2 and with all its implementations s_3.

The foregoing addresses the problem of safely modifying a conformant and interoperable player. Another interesting application is to modify a party that is known to be noninteroperable with a new partner so to make it interoperable by automatically building an appropriate patch, while preserving its interoperability with its previous partners [27]. Let us consider a party that is (and we wish to keep) *interoperable* with a set of partners, but is *noninteroperable* with a new partner. Let us also suppose to have no information about the new partner's interactive behavior. For this reason, interoperablity cannot be checked a priori but is possibly discovered during the interaction. In this case, the only information source that is available is the set of *locally observable errors* (for instance, the player is aware of messages that it receives unexpectedly). Interestingly, by exploiting the feedbacks and by using the edit operations, it is possible to *mend the player*, that we own, thus making it interoperable with another one, whose interactive behavior is actually unknown. For example, it may respond by sending back a message indicating the incoming message was not expected.

In other circumstances, the player may lack such feedback (for instance, it may observe that the communication with the partner is delayed –possibly interrupted– but it does not have any means for understanding the reason). When no feedback is available, there is no hint on how to fix the interaction and, therefore, one can only guess the modifications to try. In any case, the proposed edit operations *preserve interoperability* with all those partners the player being modified was interoperable. In other words, a player that is modified so to be able to deal with a new client will still be able to interact with all its old clients without any need of rechecking their interoperability.

In general, given a party we can produce various modifications. We define a *measure of the distance* between a party and one of its modifications, based on the number of edit operations to apply.

DEFINITION 12 (DISTANCE OF A PARTY VARIANT). *Let I be a party and let I' be another party such that $I \Rightarrow^* I'$. The* distance *between I and I' is defined as $d(I, I') = \mu n.I \Rightarrow^n I'$. We will call the shortest sequence ψ of edit operations that transforms I in I' the* shortest edit path *from I to I'.*

In this way it is possible to define patches as the modifications requiring *the least* number of changes.

DEFINITION 13 (PATCH). *Given a party I and a party T, such that I is not interoperable with T, a patch is the shortest edit path ψ that produces a service $I' \in \mathsf{update}(I)$ that is interoperable with T.*

Notice that once a patch is found that makes I interoperable with T, it is possible to further modify the upgraded party *without affecting its interoperability* with T.

5. CONCLUSION AND RELATED WORKS

This paper studies the possibility of verifying conformance and interoperability based on an explicit representation of decision points. In contrast with existing approaches, e.g., [25, 18, 5, 10], here, the interactive parties in a system can also lead choices about which message to receive and follow choices of which messages to send. Such situations arise, for instance, in blackboard systems where a partner offers a set of alternatives to its interlocutor, which chooses to accept one of them. The framework that we have proposed deals naturally with the multiparty case, while respecting our motto: *lead less, follow more!* Another advantage of the proposed approach is that it can support higher-level collaborative actions. For example, we can think of business interactions that map to message patterns involving more than one message. In such a pattern, more than one party may be a sender, but it might still be possible to distinguish who has the initiative in the pattern.

This paper explores the possibility of using the above notions not only for verification but also as a tool for automatically producing implementation replacements. The ability of producing upgrades that preserve backward compatibility is crucial, especially in contexts like internet-scale software engineering, where specifications are not always completely disclosed. The production of upgrades and of patches, aimed at gaining interoperability, is achieved through the definition of a set of transformations rules for calculating conformant variants. Such variants can be seen as conformant refinements w.r.t. the reference specification, where the notion of refinement is captured by the conformance relation. This distinguishes our work from refinement within the same agent, e.g., [4], where, in a BDI framework, subsequent levels of abstraction of a same agent are checked in order to verify that a set of target properties are preserved all along the design process. Other works, e.g., [21], consider a different kind of compliance, i.e., compliance w.r.t. Service Level Agreements. Their aim is to verify that those commitments on which the partners agreed, can actually be respected. In both cases the verifications that are performed can be considered as orthogonal to the ones proposed in our work.

The notions of conformance and interoperability have been studied by many researchers in the area of *computer aided verification* and *software engineering*. Particularly relevant is the work by Bordeaux *et al.* [8], which is set in a web service framework and assumes a *synchronous two-party* communication model. Web services are modeled as labeled transition systems (LTS) but Bordeaux *et al.*'s language does not distinguish between leading and following choices, and does not support nondeterministic choices. For this reason, their notion of conformance does not support reasoning about the advancement or postponement of decision points, as is supported by our Definition 9 (of derivability). Bordeaux *et al.* state that the extension of their approach to the nondeterministic case can easily be obtained by making nondeterministic automata deterministic by applying the classical transformation mechanisms proposed by automata theory. Unfortunately, as shown in [5], in general it is not possible to apply semantic equivalence between nondeterministic and deterministic automata: the differences in their behavior affect interoperability.

The elimination of nondeterminism by automata transformation can be used in a context where conformance is computed based on a *trace semantics* (or language containment), as in traditional approaches [26, 24, 2, 19]. The limitation of this approach is that it does not consider branching structures.

Bravetti and Zavattaro [9, 10] adopt an approach based on process algebra. Here, conformance is used as a tool for comparing service contracts [12], proposing a notion of contract refinement. Although Bravetti and Zavattaro do not distinguish between leading and following choices, this notion is close to our notion of conformance, in particular in the two-party case. The difference is that we base our definition on a variant of alternating simulation while they use testing (a trace semantics). In the multi-party case, it is necessary to make a distinction. Works like [10] are based on the assumption of dealing with web services, which, as in WSDL, simply exhibit a set of operations and must be ready to execute any of them whenever it is requested. For this reason, in [10] the addition of further receptions can be done only on channels (messages in our case) that are not present in the contract which is being refined. Our parties, instead, stick to a choreography, i.e., at every instant of

the interaction they are allowed to tackle only a subset of the messages that they are able to receive or send. The choreography also specifies who is the sender and who the receiver of each communication. Also in our case it would be possible to make a restriction analogous to [10] but the explicit specification of the leader and of the follower allows us to exploit a different kind of information, that, in the case of web services, can be found in the languages for representing choreographies.

In the context of computer aided verification, particularly relevant are the works by Rajamani and colleagues [25, 18] and de Alfaro and Henzinger *et al.* [3, 14]. The problem that we address here can be set in de Alfaro and Henzinger's [14] classification as the problem of *compositional refinement* of an implementation to a specification. Alur *et al.* [3] were first to use a simulation that deals with emissions and receptions asymmetrically. Fournet *et al.*'s approach [18] has inspired to distinguish between leading and following choices. The definition of conformance of Rajamani *et al.* preserves substitutability w.r.t. interoperability, interpreted as stuck-freeness. But their framework tackles the multiparty case only with an additional restriction that it is not possible to have more receptions than what foreseen by the specification, which we argued above is unrealistic. Rajamani *et al.* address the problem of verification but do not propose a derivation framework for producing conformant variants.

For what concerns future work, besides further exploring the research issues raised by the production of conformant upgrades and patches, we also mean to further extend the management of multiparty interactions by taking into account other information, such as causality and time, along the lines of [15]. Another interesting and open research issue is to study how to define a conformance relation that allows to capture the greatest set of conformant services, along the line of [10].

6. REFERENCES

[1] *Unified Modeling Language: Superstructure, version 2.0*, 2004.

[2] M. Alberti, D. Daolio, P. Torroni, M. Gavanelli, E. Lamma, and P. Mello. Specification and verification of agent interaction protocols in a logic-based system. In *ACM SAC 2004*, pages 72–78. ACM, 2004.

[3] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In *CONCUR*, volume 1466 of *LNCS*, pages 163–178. Springer, 1998.

[4] L. Astefanoaei and F. S. de Boer. Model-checking agent refinement. In *Proc. of AAMAS '08*, pages 705–712, 2008.

[5] M. Baldoni, C. Baroglio, A. Martelli, and V. Patti. A priori conformance verification for guaranteeing interoperability in open environments. In *Proc. of ICSOC 2006*, LNCS 4294, pages 339–351. Springer, 2006.

[6] M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella. Verifying protocol conformance for logic-based communicating agents. In *Proc. of CLIMA V*, number 3487 in LNCS, pages 192–212. Springer, 2005.

[7] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In *Proc. of Int. Conf. on Service-Oriented Computing, ICSOC'03*, LNCS, pages 43–58. Springer, 2003.

[8] L. Bordeaux, G. Salaün, D. Berardi, and M. Mecella. When are two web services compatible? In *TES 2004*, volume 3324 of *LNCS*, pages 15–28. Springer, 2005.

[9] M. Bravetti and G. Zavattaro. Contract based multi-party service composition. In *FSEN*, volume 4767 of *LNCS*, pages 207–222. Springer, 2007.

[10] M. Bravetti and G. Zavattaro. A theory for strong service compliance. In *COORDINATION*, volume 4467 of *LNCS*, pages 96–112. Springer, 2007.

[11] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: a new approach to the design and analysis of e-service composition. In *Proc. of WWW'03 Conference*, pages 403–410. ACM Press, 2003.

[12] S. Carpineti, G. Castagna, C. Laneve, and L. Padovani. A formal account of contracts for web services. In *WS-FM*, volume 4184 of *LNCS*, pages 148–162. Springer, 2006.

[13] A. K. Chopra and M. P. Singh. Interoperation in protocol enactment. In *Declarative Agent Languages and Technologies V*, volume 4897 of *LNCS*, pages 36–49, 2008.

[14] L. de Alfaro and T. A. Henzinger. Interface theories for component-based design. In *EMSOFT*, pages 148–165, 2001.

[15] N. Desai. *Interorganizational Business Interactions: Contracts, Processes, Evolution*. PhD thesis, Dep.t of Comp. Sci., North Carolina State Univ., November 2007.

[16] N. Desai, A. U. Mallya, A. K. Chopra, and M. P. Singh. Interaction protocols as design abstractions for business processes. *IEEE Transactions on Software Engineering*, 31(12):1015–1027, December 2005.

[17] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-based analysis of obligations in web service choreography. In *Proc. of IEEE International Conference on Internet&Web Applications and Services*, 2006.

[18] C. Fournet, C. A. R. Hoare, S. K. Rajamani, and J. Rehof. Stuck-free conformance. In *CAV*, volume 3114 of *LNCS*, pages 242–254. Springer, 2004.

[19] L. Giordano and A. Martelli. Verifying Agent Conformance with Protocols Specified in a Temporal Action Logic. In *Proc. of AI*IA 2007*, volume 4733 of *LNAI*, pages 145–156. Springer, September 2007.

[20] N. Guermouche, O. Perrin, and C. Ringeissen. Timed Specification For Web Services Compatibility Analysis. In *Proc. of Int. Workshop on Automated Specification and Verification of Web Systems, WWV'07*, 2007.

[21] A. Lomuscio, H. Qu, and M. Solanki. Towards verifying compliance in agent-based web service compositions. In *Proc. of AAMAS '08s*, pages 265–272, 2008.

[22] T. Miller and P. McBurney. Annotation and matching of first-class agent interaction protocols. In *Proc. of AAMAS '08*, pages 705–712, 2008.

[23] R. Milner. A calculus of communicating systems. *Lect. Notes Comp. Science*, 92, 1980.

[24] F. Plasil and S. Visnovsky. Behavior protocols for software components. *IEEE Trans. Software Eng.*, 28(11):1056–1076, 2002.

[25] S. K. Rajamani and J. Rehof. Conformance checking for models of asynchronous message passing software. In *CAV*, volume 2404 of *LNCS*, pages 166–179. Springer, 2002.

[26] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proc. of LICS, Symposium on Logic in Computer Science*, pages 332–344. IEEE Computer Society, 1986.

[27] D. M. Yellin and R. E. Strom. Protocol specifications and component adaptors. *ACM Trans. Program. Lang. Syst.*, 19(2):292–333, 1997.