A-Priori Verification of Web Services with Abduction

Marco Alberti[1], Federico Chesani[2], Marco Gavanelli[1], Evelina Lamma[1], Paola Mello[2], and Marco Montali[2]

[1] ENDIF, Università di Ferrara - Via Saragat, 1 - 44100 Ferrara (Italy).
Email: {marco.alberti|marco.gavanelli|evelina.lamma}@unife.it
[2] DEIS, Università di Bologna - Viale Risorgimento 2 - 40126 Bologna (Italy).
Email: {fchesani|pmello|mmontali}@deis.unibo.it

Abstract. Although stemming from very different research areas, Multi-Agent Systems (MAS) and Service Oriented Computing (SOC) share common topics, problems and settings. A common problem is the need to formally verify the conformance of individuals (Agents or Web Services) to common rules and specifications (resp. Protocols/Choreographies), in order to provide a coherent behaviour and to reach the user's goals.
In previous publications, we developed a framework, $\mathcal{S}$CIFF, for the automatic verification of compliance of agents to protocols. The framework includes a language based on abductive logic programming and on constraint logic programming for formally defining the social rules. Suitable proof-procedures to check on-the-fly and a-priori the compliance of agents to protocols have been defined.
Building on our experience in the MAS area, in this paper we make a first step towards the formal verification of web services conformance to choreographies in Abductive Logic Programming. We adapt the $\mathcal{S}$CIFF framework for the new settings, and propose a heir of $\mathcal{S}$CIFF, the framework A^{l}LoWS (Abductive Logic Web-service Specification). A^{l}LoWS comes with a language for defining formally a choreography and a web service specification. As its ancestor, A^{l}LoWS has a declarative and an operational semantics. We show examples of how A^{l}LoWS deals correctly with interaction patterns previously identified. Moreover, thanks to its constraint-based semantics, A^{l}LoWS deals seamlessly with other cases involving constraints and deadlines.

Note. An extended version of this paper will appear in the Proceedings of the Eighth ACM-SIGPLAN International Symposium on Principles and Practice of Declarative Programming (PPDP'06).

1 Introduction

The mating between high availability of network resources and the growing of software applications is breeding in recent years new technologies and software tools. Network ubiquity, joined together with the software engineering requirement to combine existing tools and divide the complexity of applications, is spawning new programming paradigms.

One of the most successful is Service Oriented Computing, one of the children of Object Oriented Computing. Web service technology is an important instance of Service Oriented Computing aiming at facilitating the integration of new applications, avoiding difficulties due to different platforms, languages, etc. In this context the way to build complex services from simpler ones is called *composition* and it is a very interesting and promising research area. Service composition promises to considerably reduce development time and costs by taking components off-the-shelf and joining them in a working application. Although very appealing, it leaves open questions, such as correctness of the composition, or ensuring interoperability of the web services. Choreographies propose an answer to such questions. The behaviour of the various web services is defined through a language (e.g., WS-CDL [5]), explaining the information flows amongst the components. However, the formal proofs of conformance of a web service to a choreography are not yet fully given.

Another technology descending from networks and artificial intelligence is the Multi Agent Systems (MAS). In the MAS area there exists a wide literature about checking the conformance of an agent to social rules (or protocols), both at run-time and at design-time. Baldoni et al. [4], amongst others, pointed out the similarities of requirements in the two areas of web service composition and multi agent systems. Both are devoted to define a collaboration between a collection of peers that share common goals. Both choreographies and societies should capture interactions and dependencies between interactions (time constraints, deadlines, control-flow dependencies, etc.). Both describe the external behaviour of members avoiding the internal details of the implementation of the peers.

Stemming from previous experience in multi-agent systems, we follow the path of Baldoni et al. and propose to apply agent techniques to web service composition. Within the SOCS european project [14], we proposed a formal language to define multi agent protocols [2]. We gave an abductive semantics to the devised language, and developed an abductive proof procedure, called $\mathcal{S}$CIFF [3], to check on-line the compliance of agents to protocols. More recently, we applied a variant of $\mathcal{S}$CIFF, called g-$\mathcal{S}$CIFF [1], to the problem of proving properties of a protocol itself (such as security properties). In this work, we apply these technologies to web service composition, and propose a framework, called A^{l}LoWS (Abductive Logic Web-service Specification), that exploits the variants of $\mathcal{S}$CIFF for checking the interoperability of web services.

2 The A^{l}LoWS framework

We propose an abductive based framework, A^{l}LoWS (Abductive Logic Web-service Specification), to verify conformance of web services to choreographies.

An Abductive Logic Program (ALP) [11] is a triple $\mathcal{P} \equiv \langle KB, \mathcal{E}, \mathcal{IC} \rangle$ where KB is a logic program, $\mathcal{E}$ is a set of predicates, called *abducibles*, that have no definition in KB, and $\mathcal{IC}$ is a set of *Integrity Constraints*, that must always hold true. The aim is to find a set $\Delta \subseteq \mathcal{E}$ that explains a goal G and such that the integrity constraints are satisfied

$$KB \cup \Delta \models G, \qquad KB \cup \Delta \models \mathcal{IC}.$$

A^{l}LoWS builds upon the tools and technologies developed in the SOCS project in the Multi Agent Systems area. In A^{l}LoWS, the behaviour of the web services is represented by means of *events*. Since we focus on the interactions between web services, events represent exchanged messages. We adopt the syntax used in [4]: a message is a term $m_x(Sender, Receiver, Content)$, where m_x is the type of message, and the arguments retain their intuitive meaning. We may omit some of the parameters when the meaning is clear from the context.

In A^{l}LoWS we have two types of events. Happened events are represented as $\mathbf{H}(Message, Time)$, where $Message$ has the syntax previously defined, and $Time$ is an integer, representing the time point in which the event happened. As we will see in the following, the **H** predicate can be abduced, when making hypotheses on the possible interactions. In other phases, they are considered as given a priori, thus considered as a defined predicate.

The second type of events are *expectations*. From both web services and choreography's viewpoint, given a past history of happened events, more events are expected to happen, in a conformant evolution. We represent expectations with the predicate $\mathbf{E}_X(Message, Time)$ expressing the fact that the corresponding event is expected to happen, in order to fulfil the coherent evolution, from the viewpoint of X (where X might be either the choreography or a web service). An expectation is called *fulfilled* if there exists a matching **H** event. For example, the expectation $\mathbf{E}(p(X), T)$ can be fulfilled by the event $\mathbf{H}(p(a), 1)$.

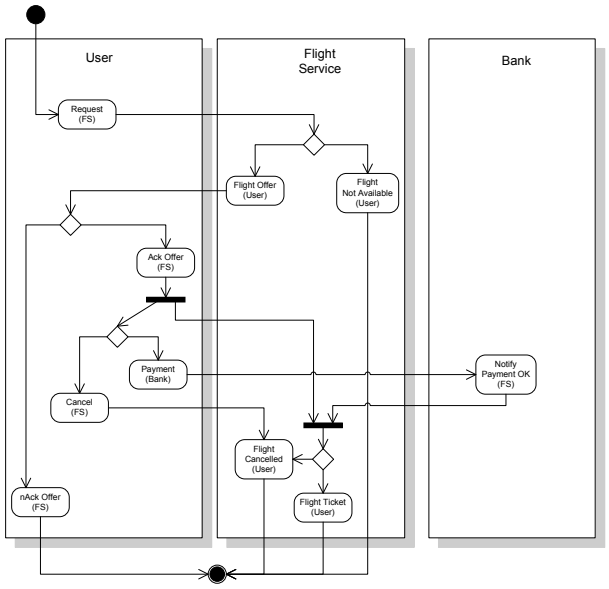

Fig. 1. Graphical representation of a simple choreography

Specification 2.1 The specification of the choreography shown in Fig. 1.

$$\mathbf{H}(request(User, FS, Flight), T_r) \rightarrow \mathbf{E}_{chor}(offer(FS, User, Flight, Price), T_o) \quad (1)$$
$$\vee\ \mathbf{E}_{chor}(notAvailable(FS, User, Flight), T_{na})$$
$$\mathbf{H}(offer(FS, User, Flight, Price), T_o) \rightarrow \mathbf{E}_{chor}(ackOffer(User, FS, Flight, Price), T_a) \quad (2)$$
$$\vee\ \mathbf{E}_{chor}(nAckOffer(User, FS, Flight, Price), T_a)$$
$$\mathbf{H}(ackOffer(User, FS, Flight, Price), T_a) \rightarrow \mathbf{E}_{chor}(payment(User, Bank, Price, FS), T_f) \quad (3)$$
$$\vee\ \mathbf{E}_{chor}(cancel(User, FS, Flight), T_f)$$
$$\mathbf{H}(ackOffer(User, FS, Flight, Price), T_a) \wedge \quad (4)$$
$$\mathbf{H}(notifyPayment(Bank, FS, Price), T_p) \rightarrow \mathbf{E}_{chor}(flightTicket(FS, User, Flight), T_f)$$
$$\vee\ \mathbf{E}_{chor}(flightCancelled(FS, User, Flight), T_f)$$
$$\mathbf{H}(cancel(User, FS, Flight), T_a) \rightarrow \mathbf{E}_{chor}(flightCancelled(FS, User, Flight), T_f) \quad (5)$$
$$\mathbf{H}(payment(User, Bank, P, Cred), T_p) \rightarrow \mathbf{E}_{chor}(notifyPayment(Bank, Cred, P), T_n) \quad (6)$$

2.1 Specification of a Choreography

A choreography describes, from a global viewpoint, the patterns of communication allowed in a system adopting such choreography [5]. The choreography specification defines the allowed messages: all messages that are not explicitly specified are forbidden. The choreography also enlists the participants, the roles the participants can play, and other knowledge about the web service interaction.

We specify a choreography by means of an ALP. A choreography specification $\mathcal{P}_{chor}$ is defined by $\mathcal{P}_{chor} \equiv \langle KB_{chor}, \mathcal{E}_{chor}, \mathcal{IC}_{chor} \rangle$. The abducibles $\mathcal{E}_{chor}$ include the happened events ($\mathbf{H}$) and the choreography's expectations ($\mathbf{E}_{chor}$).

The *Knowledge Base* (KB_{chor}) specifies declaratively pieces of knowledge of the choreography, such as roles descriptions, list of participants, etc. KB_{chor} is a set of clauses (a logic program); the clauses may contain in their body expectations about the behaviour of participants, defined literals, and constraints.

Choreography Integrity Constraints $\mathcal{IC}_{chor}$ are forward rules, of the form *Body* $\rightarrow$ *Head*, whose *Body* can contain literals and (happened, $\mathbf{H}$, and expected, $\mathbf{E}_{chor}$) events, and whose *Head* can contain (disjunctions of) conjunctions of expectations. The syntax of $\mathcal{IC}_{chor}$ is a simplified version of that defined for the $\mathcal{S}$CIFF Integrity Constraints [3]. In particular in A^{l}LoWS we do not need negative expectations and explicit negation.

In Fig. 1 a multi-party interaction is shown, expressed by the $\mathcal{IC}_{chor}$ in Spec. 2.1: the depicted scenario is about a User that wants to buy a flight ticket from a Flight Service, and pay by sending a payment order to a Bank.

CLP constraints [10] can be used to impose relations on any of the variables that occur in an expectation, like conditions on the role of the participants, or on the time instants the events are expected to happen. For example, time conditions might define orderings between the messages, or enforce deadlines.

A choreography can be *goal directed*, i.e. a specific goal $\mathcal{G}_{chor}$ can be specified: for example, a choreography used in an electronic auction system could have the goal of selling all the goods in the store. The syntax of the goal is the same as the body of a clause. If no particular goal is required, $\mathcal{G}_{chor} = true$.

Specification 2.2 The interface behaviour specification of the web service shown in Fig. 2.

$$\mathbf{H}(request(User, fs, Flight), T_r) \rightarrow \mathbf{E}_{fs}(offer(fs, User, Flight, Price), T_o) \quad (7)$$
$$\vee\ \mathbf{E}_{fs}(notAvailable(fs, User, Flight), T_{na})$$
$$\mathbf{H}(offer(fs, User, F, P), T_o) \rightarrow \mathbf{E}_{fs}(User, fs, ackOffer(User, fs, F, P), T_a) \quad (8)$$
$$\vee\ \mathbf{E}_{fs}(User, fs, nAckOffer(User, fs, F, P), T_a)$$
$$\mathbf{H}(notifyPayment(Bank, fs, Price), T_p) \rightarrow \mathbf{E}_{fs}(flightCancelled(fs, User, Flight), T_c) \quad (9)$$
$$\wedge T_p > T_a + \delta \wedge T_c > T_p$$
$$\vee\ \mathbf{E}_{fs}(flightTicket(fs, User, Flight), T_t)$$
$$\wedge T_t > T_p$$
$$\mathbf{H}(ackOffer(User, fs, Flight, Price), T_a) \rightarrow \mathbf{E}_{fs}(notifyPayment(Bank, fs, Price), T_p) \quad (10)$$
$$\vee\ \mathbf{E}_{fs}(User, fs, cancel(User, fs, Flight), T_c)$$
$$\mathbf{H}(cancel(User, fs, Flight), T_a) \rightarrow \mathbf{E}_{fs}(flightCancelled(fs, User, Flight), T_f) \quad (11)$$

2.2 Representing Web services

Similarly to the specification of a choreography, we describe the interface behaviour of a web service by means of an ALP. We restrict our analysis to the communicative aspects of the interface behaviour of a web service. A Web Service Interface Behaviour Specification $\mathcal{P}_{ws}$ is the ALP $\mathcal{P}_{ws} \equiv \langle KB_{ws}, \mathcal{E}_{ws}, \mathcal{IC}_{ws} \rangle$.

KB_{ws} and $\mathcal{IC}_{ws}$ are analogous to their counterparts in the choreography specification, except that they are an individual, rather than global, perspective: they represent the declarative knowledge and the policies of the web service.

$\mathcal{E}_{ws}$ is the set of abducible predicates: as for the choreographies, it contains both expectations and happened events. The expectations in $\mathcal{E}_{ws}$ can be divided into two significant subsets:

- expectations about messages whose sender is ws, $\mathbf{E}_{ws}(m_x(ws, A, Content))$, are interpreted as actions that ws intends to do;
- expectations about messages uttered by other participants to ws (of the form $\mathbf{E}_{ws}(m_x(A, ws, Content))$, with $A \neq ws$), can be intended as the messages that ws is able to understand.

In Fig. 2 the communicative part of a web service's interface behaviour is represented. The corresponding translation in terms of $\mathcal{IC}_{ws}$ is in Spec. 2.2.

3 Conformance: declarative semantics

Intuitively, conformance is the characteristics of a web service to comply to a choreography, provided that the other peers will behave according to the choreography. From the declarative semantics viewpoint, the test of conformance requires to assume further hypotheses about events ws expects to utter, and events that the choreography expects other peers to utter. We consider the predicate $\mathbf{H}$ as abducible, and use the web service's interface behaviour $\mathcal{P}_{ws}$ to foresee the messages the web service will send in every possible situation, provided that the other peers behave as specified by the choreography. Formally, all the messages the web service ws expects to send will be delivered, i.e.:

$$\mathbf{E}_{ws}(m_x(ws, R, C), T) \rightarrow \mathbf{H}(m_x(ws, R, C), T) \quad (12)$$

Symmetrically, the messages sent by other peers are those prescribed by the choreography specification $\mathcal{P}_{chor}$:

$$\mathbf{E}_{chor}(m_x(S,R,C),T), S \neq ws \rightarrow \mathbf{H}(m_x(S,R,C),T) \tag{13}$$

The possible interactions amongst the web service ws and the other peers will be the sets $\mathbf{HAP}^*$ satisfying equations 12 and 13.

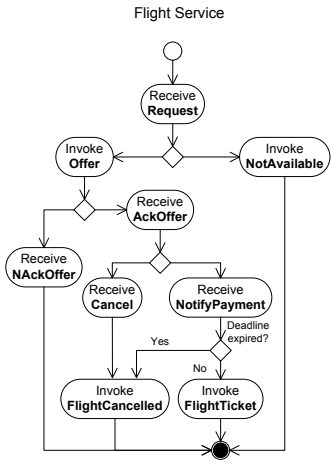

Fig. 2. Example of a behavioural interface

Definition 1. *Given the ALP* $\langle KB_U, \mathcal{E}_U, \mathcal{IC}_U \rangle$ *(where* $KB_U \triangleq KB_{chor} \cup KB_{ws}$, $\mathcal{E}_U \triangleq \mathcal{E}_{chor} \cup \mathcal{E}_{ws}$, $\mathcal{IC}_U \triangleq \mathcal{IC}_{chor} \cup \mathcal{IC}_{ws}$*), a* possible interaction *of a web service* ws *in a choreography* $chor$ *is a pair* $(\mathbf{HAP}^*, \mathbf{EXP}^*)$ *such that:*

$$KB_U \cup \mathbf{HAP}^* \cup \mathbf{EXP}^* \models G_U \tag{14}$$

$$KB_U \cup \mathbf{HAP}^* \cup \mathbf{EXP}^* \models \mathcal{IC}_U \tag{15}$$

$$KB_U \cup \mathbf{HAP}^* \cup \mathbf{EXP}^* \models (12) \cup (13) \tag{16}$$

(where by Eq. 16 we mean that equations 12 and 13 must hold). The set $\mathbf{HAP}^*$ *is also called* possible history.

When the goal G_U is *true*, the empty set is typically one of the possible histories. The empty history is often of little (or no) interest for proving conformance. When the interesting histories are only those containing at least one event, the expectation of such event can be inserted as the goal G_U. Typically, we use as goal the expectation (both from the web service's viewpoint, $\mathbf{E}_{ws}$, and from the choreography's viewpoint, $\mathbf{E}_{chor}$) of the first event of an interaction. This poses no serious restriction on the types of protocols that can be tested.

Example 1. Suppose a choreography prescribes the following protocol:

$$\mathbf{H}(ask(ws, R, X)) \rightarrow \mathbf{E}_{chor}(answer(R, ws, X)) \tag{17}$$
$$\mathbf{H}(answer(R, ws, X)) \rightarrow \mathbf{E}_{chor}(ack(ws, R, X)) \tag{18}$$

while the web service's integrity constraints contain only the first rule

$$\mathbf{H}(ask(ws, R, X)) \rightarrow \mathbf{E}_{ws}(answer(R, ws, X)).$$

Let $G_U = \mathbf{E}_{ws/chor}(ask(ws, peer, X))$.[3] Given the goal G_U, ws has the intention to send an *ask* message to the *peer*, so all the possible histories for G_U will contain the event $\mathbf{H}(ask(ws, peer, X))$. The *peer*'s behaviour is simulated through the rules in the choreography specification. Since the choreography has an expectation (generated by rule 17) $\mathbf{E}_{chor}(answer(peer, ws, X))$, this will become a happened event: $\mathbf{H}(answer(peer, ws, X))$. Now, rule 18 provides a choreography's expectation about the third message: ws is supposed to send an *ack*. But, as we can see from ws's specification, it does not have an expectation to send such message, so the simulation will not suppose it will comply to the choreography's expectation. So, the (only) possible history for the goal G_U is

$$\mathbf{HAP}^* = \{\mathbf{H}(ask(ws, peer, X)), \mathbf{H}(answer(peer, ws, X))\}. \tag{19}$$

In a possible history, messages sent by the other peers comply by definition to the choreography. However, the messages uttered by the web service ws under test might be non conformant. ws is conformant if all the possible histories are conformant. Also, ws should be able to understand the messages it receives, otherwise there might be requests which it is unable to serve. We require all possible histories to satisfy both the choreography and the web service expectations.

Definition 2. *A possible history* $\mathbf{HAP}^*$ *is* Feeble Conformant *if there exists a set* $\mathbf{EXP}$ *such that*[4]

$$KB_U \cup \mathbf{HAP}^* \cup \mathbf{EXP} \models G \tag{20}$$
$$KB_U \cup \mathbf{HAP}^* \cup \mathbf{EXP} \models \mathcal{IC}_U \tag{21}$$
$$\mathbf{HAP}^* \cup \mathbf{EXP} \models \mathbf{E}_{ws}(X) \rightarrow \mathbf{H}(X) \tag{22}$$
$$\mathbf{HAP}^* \cup \mathbf{EXP} \models \mathbf{E}_{chor}(X) \rightarrow \mathbf{H}(X) \tag{23}$$

A web service is feeble conformant if all the possible histories are feeble conformant. A pair $(\mathbf{HAP}^*, \mathbf{EXP})$ *is a* Feeble Conformant Interaction *if* $\mathbf{HAP}^*$ *is a feeble conformant history and* $\mathbf{EXP}$ *is a set of expectations satisfying equations (20-23) which is minimal with respect to set inclusion.*

Example 2. Cont. from Example 1. In the history of Eq. 19, the choreography's expectation for the message *ask* is unfulfilled, so ws is not (feeble) conformant.

[3] $\mathbf{E}_{ws/chor}$ represents an event expected both by the choreography and the web service.

[4] Note the difference between Eq. 22-23 and Eq. 12-13: Eq. 22 is used as a test, and requires *all* the expectations of the web service to be fulfilled, while Eq. 12 is used to *generate* the behaviour of the web service and imposes only the fulfilment of the expectations the web service has about *itself*. Analogously for Eq. 23 and 13.

Feeble conformance ensures that ws will utter all the messages requested by the choreography, but it does not require ws to avoid the forbidden messages. We extend feeble conformance to a stronger version.

A possible history is strong conformant if (it is feeble conformant and) all the happened events were expected both by the choreography and the web service. We include in this concept only the communications that involve the web service under observation (the other messages, exchanged between other peers in a multi-party interaction, are considered conformant).

Definition 3. *A feeble conformant interaction* (**HAP***, **EXP**) *is also a* Strong Conformant Interaction *if the following conditions hold:*

$$\mathbf{H}(m_x(ws, R, C)) \leftrightarrow \mathbf{E}_{chor}(m_x(ws, R, C)) \tag{24}$$
$$\mathbf{H}(m_x(S, ws, C)) \leftrightarrow \mathbf{E}_{ws}(m_x(S, ws, C)). \tag{25}$$

A Strongly Conformant History *is a history for which there exists a strongly conformant interaction. A web service is* Strongly Conformant *if all the possible histories are strongly conformant.*

Example 3. Let us change in the previous example the specifications of the choreography and of the web service, i.e., the web service specification is

$$\mathbf{H}(ask(ws, R, X)) \rightarrow \mathbf{E}_{ws}(answer(R, ws, X))$$
$$\mathbf{H}(answer(R, ws, X)) \rightarrow \mathbf{E}_{ws}(ack(ws, R, X))$$

and the choreography is

$$\mathbf{H}(ask(ws, R, X)) \rightarrow \mathbf{E}_{chor}(answer(R, ws, X)).$$

ws intends to send ack, so it will indeed send it in all the possible histories. The choreography does not prescribe this message. The possible history is

$$\mathbf{HAP}^*{}_2 = \{\mathbf{H}(ask(ws, peer, X)), \mathbf{H}(answer(peer, ws, X)), \mathbf{H}(ack(ws, peer, X))\}.$$

All expectations of the choreography are fulfilled by some message of ws, so ws is feeble conformant. However, it will also send an unrequested message, that might confound the other $peer$, undermining the interoperability. The ack message is unexpected by the choreography, therefore ws is not strong conformant.

3.1 Operational semantics

A^{l}LoWS operational semantics is based on the two versions of the $\mathcal{S}$CIFF proof procedure developed in the SOCS project. $\mathcal{S}$CIFF is sound and complete; it terminates for acyclic programs. The $\mathcal{S}$CIFF proof procedure considers the **H** events as a predicate defined by a set of incoming atoms; it is devoted to generate expectations corresponding to such history and to check that expectations indeed match with happened events. $\mathcal{S}$CIFF was developed to check the compliance of agents to protocols [3]. The $\mathcal{S}$CIFF proof procedure is based on a rewriting system transforming one node to another (or to others) as specified by rewriting steps called *transitions*. A node is defined by the

tuple $T \equiv \langle R, CS, PSIC, \mathbf{PEND}, \mathbf{HAP}, \mathbf{FULF}, \mathbf{VIOL} \rangle$, where R is the resolvent, CS is the constraint store à la CLP [10], $PSIC$ is a set of implications, derived from the $\mathcal{IC}$s, **HAP** is the history of happened events, and the set of expectations is partitiond into **PEND** (pending), **FULF** (fulfilled) and **VIOL** (violated). We cannot report here all the transitions, due to lack of space. As an example, the *fulfilment* transition is devoted to prove that an expectation $\mathbf{E}(X, T_x)$ has been fulfilled by an event $\mathbf{H}(Y, T_y)$. Two nodes are generated: in the first, X and Y are unified, and the expectation is fulfilled (i.e., it is moved to the set **FULF**); in the second the new constraint $X \neq Y$ is added to the constraint store CS. At the end of the computation, a *closure* transition is applied, and all the expectations remaining in the set **PEND** are considered as violated.

The g-$\mathcal{S}$CIFF proof procedure, instead, considers **H** an abducible predicate and provides both the set of expectations and the history that fulfils the goal. g-$\mathcal{S}$CIFF was used to prove properties of protocols, such as security [1]. It has the same transitions in $\mathcal{S}$CIFF; in the version adopted in this paper, it also contains as integrity constraints the rules 12 and 13. g-$\mathcal{S}$CIFF generates events, so it does not contain the *closure* transition, that enforces a closed world assumption on the set of happened events. A derivation without *closure* is called *open*.

In order to prove conformance, we apply the two proof procedures to the two phases implicitly defined in the previous section. We decompose the proof of feeble conformance into a *generative* phase and a *test* phase. In the generative phase, we generate, by means of g-$\mathcal{S}$CIFF, all the possible histories. Of course, those histories need not be generated as ground histories (the set of ground histories can be infinite), but intensionally: the **H** events can contain variables, possibly with constraints à la Constraint Logic Programming (CLP) [10].

In the test phase, we check with $\mathcal{S}$CIFF the compliance of the generated histories both with respect to the web service and the choreography specifications. If all the histories are conformant, the web service is feeble conformant to the choreography. Otherwise, if there exists at least one history that is not conformant, the web service is not (feeble) conformant.

Finally, we can prove strong conformance by checking that all the happened events were expected both by the choreography and by the web service. This can be performed during the second phase ($\mathcal{S}$CIFF) by adopting the same technique used in the fulfilment transition: if a **H** event matches both with an $\mathbf{E}_{ws}$ and a $\mathbf{E}_{chor}$ expectation, it is labelled *expected*; after the application of the *closure* transition, all events that were not expected are considered *unexpected*, showing that the web service was not strongly conformant.

3.2 Examples

Baldoni et al. [4] show various examples of conformance and non-conformance of a web service to a choreography, and propose a framework based on Finite State Automata, to prove conformance. We show how their examples are addressed in A^{l}LoWS, based on Computational Logics.

Web service with more capabilities In the first example of [4], the choreography specification defines only one allowed interaction: ws sends a message m_1 and the other peer will reply with m_2 (Eq. 26). The specification of ws is wider:

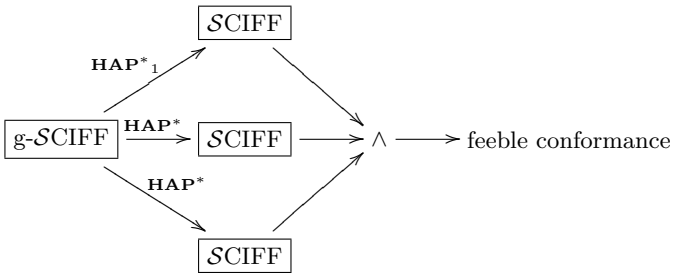

Fig. 3. A^{l}LoWS architecture

after m_1, ws accepts either m_2 or m_3 (Eq. 27).

$$\mathbf{H}(m_1(ws, X)) \rightarrow \mathbf{E}_{chor}(m_2(X, ws)) \tag{26}$$

$$\mathbf{H}(m_1(ws, X)) \rightarrow \mathbf{E}_{ws}(m_2(X, ws)) \vee \mathbf{E}_{ws}(m_3(X, ws)) \tag{27}$$

Baldoni et al. state that ws is conformant. In fact, in a legal conversation message m_3 will never be received by ws, so the interoperability is ensured.

The g-$\mathcal{S}$CIFF proof procedure is started with the goal containing the expectation, both from the web service's and from the choreography's viewpoint, of the first event: $G_U = \mathbf{E}_{ws/chor}(m_1(ws, X))$. g-$\mathcal{S}$CIFF abduces (Fig. 4) one possible history: $\mathbf{HAP}^* = \{\mathbf{H}(m_1), \mathbf{H}(m_2)\}$. There are two alternative sets of expectations from the viewpoint of ws, $\{\mathbf{E}_{ws}(m_1), \mathbf{E}_{ws}(m_2)\}$ and $\{\mathbf{E}_{ws}(m_1), \mathbf{E}_{ws}(m_3)\}$, but in the first phase the correspondence between expectations and happened events is not required (open derivation). In the second phase, the (only) generated history is checked; since there exists one set of expectations fulfilled by $\mathbf{HAP}^*$, ws is feeble conformant. Since in $\mathbf{HAP}^*$ there are no unexpected events, ws is also strong conformant.

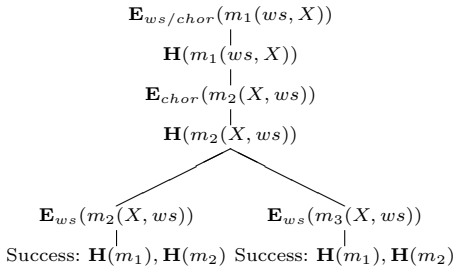

Fig. 4. g-$\mathcal{S}$CIFF derivation.

Missing capability The web service accepts as reply only m_2 (Eq 28), while the choreography defines as valid two interactions (Eq 29):

$$\mathbf{H}(m_1(ws, X)) \rightarrow \mathbf{E}_{ws}(m_2(X, ws)) \tag{28}$$

$$\mathbf{H}(m_1(ws, X)) \rightarrow \mathbf{E}_{chor}(m_2(X, ws)) \vee \mathbf{E}_{chor}(m_4(X, ws)). \tag{29}$$

In this case, g-$\mathcal{S}$CIFF provides two possible histories: $\mathbf{HAP^*}_1 = \{\mathbf{H}(m_1), \mathbf{H}(m_2)\}$ and $\mathbf{HAP^*}_2 = \{\mathbf{H}(m_1), \mathbf{H}(m_4)\}$. In phase 2, $\mathcal{S}$CIFF detects non conformance of $\mathbf{HAP^*}_2$, because the expectation $\mathbf{E}_{ws}(m_2(S, ws, C))$ remains unfulfilled in all possible derivation paths. This means that ws is blocked waiting for m_2, and will not process other messages, so it is not (feeble) conformant.

Wrong reply ws assumes to have the freedom to reply either m_2 or m_3 to a question m_1 (Eq. 30), while the choreography does not grant such a freedom: only m_2 is legal (Eq. 31):

$$\mathbf{H}(m_1(X, ws)) \rightarrow \mathbf{E}_{ws}(m_2(ws, X)) \vee \mathbf{E}_{ws}(m_3(ws, X)) \tag{30}$$

$$\mathbf{H}(m_1(X, ws)) \rightarrow \mathbf{E}_{chor}(m_2(ws, X)). \tag{31}$$

This case is non conformant according to [4], as in some cases ws might utter the forbidden message m_3. g-$\mathcal{S}$CIFF abduces two possible histories ($\{\mathbf{H}(m_1), \mathbf{H}(m_2)\}$ and $\{\mathbf{H}(m_1), \mathbf{H}(m_3)\}$). The first is compliant, according to $\mathcal{S}$CIFF, while in the second the choreography's expectation $\mathbf{E}_{chor}(m_2)$ remains pendent.

Predefined answer The dual of the previous example is when the choreography lets the web service choose to reply m_2 or m_3 to a question m_1 (Eq 32), while the web service sticks to the reply m_2 (Eq 33). Again, A^{l}LoWS provides a correct proof: g-$\mathcal{S}$CIFF gives one possible history $\{\mathbf{H}(m_1), \mathbf{H}(m_2\}$, which is reported (feeble and strong) conformant by $\mathcal{S}$CIFF in the second phase.

$$\mathbf{H}(m_1(X, ws)) \rightarrow \mathbf{E}_{chor}(m_2(ws, X)) \vee \mathbf{E}_{chor}(m_3(ws, X)) \tag{32}$$

$$\mathbf{H}(m_1(X, ws)) \rightarrow \mathbf{E}_{ws}(m_2(ws, X)). \tag{33}$$

Thus, in all the examples by Baldoni et al., A^{l}LoWS provides the same answer proposed in [4]. We now propose other examples that highlight the enhanced expressive power provided by computational logics, in particular the use of constraints, that are embedded in the $\mathcal{S}$CIFF and g-$\mathcal{S}$CIFF proof procedures.

Mutual exclusion Many protocols include mutual exclusion between choices: a choreography might prescribe that if a given condition on a message m_1 holds, a message m_2 should be exchanged, otherwise another message m_3 should be sent. In A^{l}LoWS, conditions can be expressed by means of constraints (either the ones predefined in the underlying solver, i.e., CLP(FD), or user-defined) or by means of defined predicates. As a simple example, suppose the choreography prescribes to reply either m_2 or m_3, depending on the content of the previous message m_1 (Eq. 34) while the web service always replies m_2 (Eq. 35):

$$\begin{aligned}\mathbf{H}(m_1(X, ws, C)) \rightarrow\ & \mathbf{E}_{chor}(m_2(ws, X, C_2)), C > 0 \\ \vee\ & \mathbf{E}_{chor}(m_3(ws, X, C_3)), C \leq 0\end{aligned} \tag{34}$$

$$\mathbf{H}(m_1(X, ws, C)) \rightarrow \mathbf{E}_{ws}(m_2(ws, X, C_2)) \tag{35}$$

g-$\mathcal{S}$CIFF generates two possible histories, with variables and constraints upon the variables (Fig 5). In both the messages m_1 and m_2 are generated, but while in the first the proof procedure assumes that C takes a value greater than 0, in the second C is non positive. In the second phase, $\mathcal{S}$CIFF takes as input both the happened events and the constraint store, and accepts as conformant the first history, while discarding as non-conformant the second.

Notice that constraints scope is not restricted only to variables in the content, but might involve all the variables in the message, including *time*.

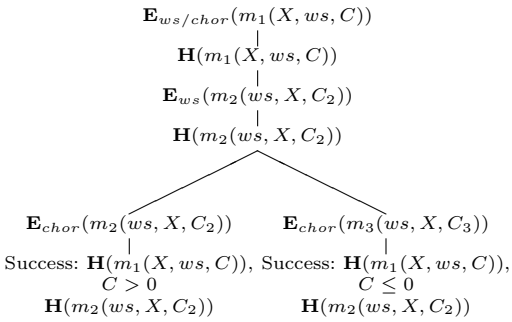

Fig. 5. g-$\mathcal{S}$CIFF derivation for Example of Mutual Exclusion.

Deadlines Suppose that the choreography specifies a deadline for the receipt of a given message m_2 (Eq. 36), and that the web service ws replies within a deadline that might be different (Eq. 37):

$$\mathbf{H}(m_1(X, ws, C_1), T_1) \rightarrow \mathbf{E}_{chor}(m_2(ws, X, C_2), T_2) \wedge T_2 < T_1 + \delta_{chor} \quad (36)$$

$$\mathbf{H}(m_1(X, ws, C_1), T_1) \rightarrow \mathbf{E}_{ws}(m_2(ws, X, C_2), T_2) \wedge T_2 < T_1 + \delta_{ws}. \quad (37)$$

In this case, the only possible history is

$$\mathbf{HAP}^* = \{\, \mathbf{H}(m_1(X, ws, C_1), T_1), \mathbf{H}(m_2(ws, X, C_2), T_2), T_2 < T_1 + \delta_{ws}\}$$

Applying $\mathcal{S}$CIFF to the history $\mathbf{HAP}^*$, generates the choreography's expectation $\mathbf{E}_{chor}(m_2(ws, X, C_2), T_2) \wedge T_2 < T_1 + \delta_{chor}$; this expectation matches with the second item of $\mathbf{HAP}^*$ if a further condition holds: $T_2 < T_1 + \delta_{chor}$. Coherently with the philosophy of CLP, $\mathcal{S}$CIFF provides this constraint in output, as a conditional answer: the web service is conformant provided that the answer arrives before the deadline in the choreography specification.

4 A test conformance example

Consider the choreography specification in Fig. 1. The interaction is initiated by a $User$ that asks the Flight Service FS to book a flight. If there are seats available on the plane, FS will reply with *flightOffer*, specifying the $Price$. Otherwise, FS replies with *notAvailable*. The *offer* can be accepted (*ackOffer*) or refused (*nAckOffer*) by the $User$. If the *offer* is accepted, FS will book the

seat. After booking, the $User$ can still $Cancel$ the reservation. Otherwise, it will issue a payment order ($payment$) to the $Bank$, that will send the notification ($notifyPayment$) to the creditor, FS. When FS has received both the booking order ($ackOffer$) and the payment ($notifyPayment$), it will normally issue the $flightTicket$ to the $User$; however, FS retains the right to refuse the ticket and send a $flightCancelled$ message in case of problems (e.g., overbooking).

Fig. 2 shows the behavioural interface of a Flight Server web service; the specification in terms of ICs is in Spec. 2.2. The FS establishes that the late payment is an error condition, and will cancel the booking if the payment notification does not arrive within δ time units after the booking.

In next section, we show how the conformance of *fs* is proven in A^{l}LoWS.

4.1 Conformance of the Flight Service

The test of conformance of the Flight Service fs is performed by generating, through g-$\mathcal{S}$CIFF, the set of the possible histories. The g-$\mathcal{S}$CIFF derivation provides five possible histories:

$$\begin{aligned}
\mathbf{HAP^*}_1 = \{&\mathbf{H}(request(U, fs, F), T_r), \mathbf{H}(offer(fs, C, F, P), T_o), \mathbf{H}(ackOffer(C, fs, F, P), T_a),\\
&\mathbf{H}(payment(C, B, P, fs), T_p), \mathbf{H}(notifyPayment(B, fs, P)), T_n) \wedge T_n > T_a + \delta\\
&\mathbf{H}(flightCancelled(fs, C, F), T_c)\},\\
\mathbf{HAP^*}_2 = \{&\mathbf{H}(request(U, fs, F), T_r), \mathbf{H}(offer(fs, C, F, P), T_o), \mathbf{H}(ackOffer(C, fs, F, P), T_a),\\
&\mathbf{H}(payment(C, B, P, fs), T_p), \mathbf{H}(notifyPayment(B, fs, P)), T_n) \wedge T_n \leq T_a + \delta\\
&\mathbf{H}(flightTicket(fs, C, F), T_t)\},\\
\mathbf{HAP^*}_3 = \{&\mathbf{H}(request(U, fs, F), T_r), \mathbf{H}(offer(fs, C, F, P), T_o), \mathbf{H}(ackOffer(C, fs, F, P), T_a),\\
&\mathbf{H}(cancel(C, fs, F), T_c), \mathbf{H}(flightCancelled(fs, C, F))\},\\
\mathbf{HAP^*}_4 = \{&\mathbf{H}(request(U, fs, F), T_r), \mathbf{H}(offer(fs, C, F, P), T_o), \mathbf{H}(nAckOffer(C, fs, F, P), T_a)\},\\
\mathbf{HAP^*}_5 = \{&\mathbf{H}(request(U, fs, F), T_r), \mathbf{H}(notAvailable(fs, C, F, P), T_n)\}.
\end{aligned}$$

Two of the histories include time constraints. All the possible histories are trivially conformant: they satisfy both the expectations of the choreography, and those of the web service fs. Thus, fs is feeble conformant. Moreover, all the generated events are expected, and this shows that fs is also strong conformant.

5 Related Work

A number of languages for specifying service choreographies and testing "a priori" and/or "run-time" conformance have been proposed in the literature. Two examples of these languages are state machines [6] and Petri nets [8].

In [6], the authors focus on two-party choreographies involving a requester and a provider (named service conversations) and formulate some requirements for a modelling language suitable for them. The requirements include genericity, automated support, and relevance. The authors argue that state machines satisfy these requirements and sketch an architecture of a service conversation controller capable of monitoring messages exchanged between a requester and provider in order to determine whether they conform to a conversation.

An example of use of Petri nets for the formalization of choreographies is in [8]. Four different viewpoints (interface behaviour, provider behaviour, choreography, and orchestration) and relations between viewpoints are identified and

formalised. These relations are used to perform (global) consistency checking of multi-viewpoint service designs thereby providing a formal foundation for incremental and collaborative approaches to service-oriented design. Our proposal is limited to a deep analysis of the relation between choreographies and behaviour interfaces but deal with both "a priori" and "run-time" conformance.

Our work is highly inspired by Baldoni et al. [4]. We adopt, like them, a MAS viewpoint, in defining a priori conformance in order to guarantee interoperability. As in [4], we interpret a-priori conformance as a property that relates two formal specifications: the global one determining the conversations allowed by the choreography and the local one related to the single web service. But, while they represent choreographies as finite state automata, we claim that the formalisms and technologies developed in the area of Computational Logic in providing a declarative representation of the social knowledge, could provide a higher expressive power. This paper can be considered as a first step in this direction. We also manage concurrency, which they do not consider at the moment.

Endriss et al. [9] apply a formalism based on computational logic to the a-priori conformance in the MAS field. They restrict their analysis to the so-called *shallow protocols*. They address only 2-party interactions, without the possibility of expressing conditions over the content of the exchanged messages, and without considering concurrency. While [9] and our work agree on the notion of strong/exhaustive conformance, we have dual notions of feeble/weak conformance: in [9] weak conformant is an agent that does not perform forbidden actions, but might fail to perform required actions. Dually, in this work, we call feeble conformant an agent that executes all the required actions, but might also perform forbidden actions.

Abduction has been applied to verification in other work; in particular, Russo et al. [13] use an abductive proof procedure for analysing event-based requirements specifications. They use abduction for analysing the correctness of specifications, while A^{l}LoWS is focussed on conformance checking of web services.

In [12], the authors tackle the problem of verifying (general and specific) properties of a service obtained from the composition of many web services. Each web service specification (written in BPEL4WS) is translated into a *labelled state transition system* (labelled STS); then, by applying a composition operator, they get the STS representing the composed service. Finally, model checking techniques are applied to this latter model, to the end of verifying the properties. We share the same intuition that *"the situation where some messages can be emitted without being ever consumed should not occur in valid composition"*. Our approach consists in representing both the web service specification and the choreography with the same logic-based formalism, and then interoperability is verified, before actual composition.

6 Conclusions and Future Work

This paper represents a first step in checking the conformance of web services to choreographies in computational logics. We proposed a framework, called A^{l}LoWS, for defining web services and choreographies specifications, and checking their conformance. We showed various examples of the expressivity of com-

putational logics, including reasoning on constraints, deadlines, and other structures that are not currently easily addressed by classical tools.

We are aware that many issues should be addressed in the future. For example, A^ILoWS generates the possible histories, though in intensional version, so it cannot currently handle histories of unbounded length, such as those possible in choreographies containing *cycles*. In such a case, the resulting program could be non acyclic, so the proof of termination might not hold. Those choreographies might be tested using an iterative deepening search strategy.

We are currently developing a graphical language to define choreographies, and a compiler to generate integrity constraints from the graphical specification.

Acknowledgments
This work has been partially supported by the MIUR PRIN 2005 projects *Specification and verification of agent interaction protocols*, and *Vincoli e preferenze come formalismo unificante per l'analisi di sistemi informatici e la soluzione di problemi reali*.

References

1. M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Security protocols verification in Abductive Logic Programming: a case study. In A. Pettorossi, M. Proietti, and V. Senni, editors, *CILC 2005*, June 21-22 2005.
2. M. Alberti, A. Ciampolini, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. A social ACL semantics by deontic constraints. In V. Mařík, J. Müller, and M. Pěchouček, editors, *CEEMAS 2003*, volume 2691 of *LNAI*, pages 204–213.
3. M. Alberti, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. The $\mathcal{S}$CIFF abductive proof-procedure. In *AI*IA 2005*, volume 3673 of *LNAI*, pages 135–147.
4. M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella. Verifying the conformance of web services to global interaction protocols: A first step. In Bravetti et al. [7].
5. A. Barros, M. Dumas, and P. Oaks. A critical overview of the web services choreography description language (WS-CDL). *BPTrends*, 2005.
6. B. Benattallah, F. Casati, F. Toumani, and R. Hamadi. Conceptual modeling of web service conversations. 2681:449–467, 2003.
7. M. Bravetti, L. Kloul, and G. Zavattaro, editors. volume 3670 of *Lecture Notes in Computer Science*. Springer, 2005.
8. R. Dijkman and M. Dumas. Service-oriented design: A multi-viewpoint approach. *International Journal of Cooperative Information Systems*, 13(4):337–378, 2004.
9. U. Endriss, N. Maudet, F. Sadri, and F. Toni. Logic-based agent communication protocols. In F. Dignum, editor, *Advances in Agent Communication*, volume 2922 of *LNAI*, pages 91–107. Springer-Verlag, 2004.
10. J. Jaffar, M. Maher, K. Marriott, and P. Stuckey. The semantics of constraint logic programs. *Journal of Logic Programming*, 37(1-3):1–46, 1998.
11. A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive Logic Programming. *Journal of Logic and Computation*, 2(6):719–770, 1993.
12. R. Kazhamiakin and M. Pistore. A parametric communication model for the verification of BPEL4WS compositions. In Bravetti et al. [7], pages 318–332.
13. A. Russo, R. Miller, B. Nuseibeh, and J. Kramer. An abductive approach for analysing event-based requirements specifications. In P. Stuckey, editor, *ICLP 2002*, volume 2401 of *LNCS*, pages 22–37. Springer-Verlag, 2002.
14. Societies Of ComputeeS (SOCS). http://lia.deis.unibo.it/Research/SOCS/.