# *Verifying Agent Conformance with Protocols: an Automata Based Approach*

Laura Giordano[1] and Alberto Martelli[2]

[1]Dipartimento di Informatica, Università del Piemonte Orientale, Alessandria

[2]Dipartimento di Informatica, Università di Torino, Torino

## Summary of the paper

- The paper addresses the problem interoperability of software agents in an open environment.
- To guarantee that an agent can correctly interact with other agents according to a given protocol, we define a notion of conformance of agents with protocols.
- We assume that the specification of interaction *protocols* is given in an action theory by means of temporal constraints, in a dynamic temporal logic.
- The verification of conformance of agents with protocols can then be done by making use of automata-based techniques.

## Domain description

A protocol is specified by a *domain description*, i.e. a pair $(\Pi, \mathcal{C})$, where

- $\Pi$ is an action theory containing:
  - *action laws*
  - *causal laws*
  - *precondition laws*
  - *initial state*
- $\mathcal{C}$ is a set of constraints, arbitrary temporal formulas of DLTL.

We can deal with terminating and infinite protocols.

## Action laws $\mathcal{AL}$

Some action laws

$\Box([sendRequest(C,P)]requested)$

$\Box([sendOffer(P,C)]CC(P,C,accepted,goods))$

$\Box(requested \rightarrow [sendOffer(P,C)]\neg requested)$

$\Box(requested \rightarrow [sendNotAvail(P,C)]\neg requested)$

$\Box([sendAccept(C,P)](accepted \wedge CC(C,P,goods,paid)))$

$\Box([sendRefuse(C,P)]\neg accepted$

$\Box([sendGoods(P,C)]goods$

$\Box([sendPayment(C,P)]paid)$

Actions are assumed to have *deterministic effects*, and states are complete sets of fluents litterals.

In the *initial state*, all fluents are taken to be false.

## Permissions $\mathcal{PL}$

The permissions to execute communicative actions in each state are represented by *precondition laws*.

The *precondition laws* for the actions of the customer are the following ones:

$$\Box(\neg Offer \rightarrow [sendAccept(C,P)]\bot)$$
$$\Box(\neg Offer \rightarrow [sendRefuse(C,P)]\bot)$$
$$\Box(\neg goods \rightarrow [sendPayment(C,P)]\bot).$$

The customer may send an accept or refuse only if an offer has been done.

The customer may send a payment for the goods only if he has received the goods.

All other actions are always executable for the customer.

## Constraints $\mathcal{C}$

Constraints in $\mathcal{C}$ are arbitrary temporal formulas of DLTL.

Examples:

$\neg\Diamond < sendOffer + sendNotAvail > \Diamond < sendOffer + sendNotAvail > \top$

For each commitment $C(i, j, \alpha)$, $\mathcal{C}$ contains the constraint:

$$\Box(C(i, j, \alpha) \rightarrow \Diamond\alpha)$$

All *commitments have to be fulfilled*.

## Conformance

Given:

- a protocol $P$ with two roles $i$ and $j$.
- an agent $S_i$ playing the role of $i$

we want to define a notion of conformance of $S_i$ with the protocol $P$.

We want to guarantee interoperability:
the interactions of $S_i$ with the other agent $S_j$ conformant with $P$ gives rise to *runs of the protocol* and produces *no deadlock* situations.

**Observe that**: The two agents share the same actions.

## Conformance of agent $S_i$ with protocol $P$

We can consider an agent $S_i$ to be *conformant* with the protocol:

- $S_i$ is *not forced to send all the messages* that could be sent according to the protocol.

- $S_i$ *can receive more messages* than those it should actually receive according to the protocol.

To summarize, a conformant agent may have *"less emissions and more reception"*.

## Conformance

We define a notion of conformance of an agent with respect to a protocol by comparing the runs of the agents and the runs of the protocol.
In particular, in this definition, we will not consider the value of fluents at the different worlds in the runs, but only the sequences of actions that can be executed according to the protocol and to the agent.

## Conformance

Informally, an agent $S_i$ conforms with a protocol $P$ if the following conditions hold:

(i) The messages sent from $S_i$ are *correct*: that is, if $S_i$ sends a message $m$ at some stage, then, the role $i$ of the protocol can send message $m$ at that stage.

(ii) $S_i$ must receive all the messages which it could receive according to the protocol. This is a *completeness* requirement for $S_i$.

(iii) If, in a state of the protocol, role $i$ is expected to send a message, in the corresponding state of agent $S_i$, it must send at least a message.

Condition (iii) is required to avoid deadlock situations when the two agents $S_i$ and $S_j$ interact: they cannot be both waiting to receive a message (*no deadlock*).

## Conformance

An agent $S_i$ is *conformant* with a protocol $P$ if, whenever there are two runs, $\sigma_S$ of $S_i$ and $\sigma_P$ of $P$, with a common prefix $\pi$, the following conditions are satisfied:

(1) if the action $send_{i,j}$ is executed after the prefix $\pi$ in $\sigma_S$, then there exist a run $\sigma$ common to $P$ and $S_i$ with prefix $\pi send_{i,j}$;

(2) if the action $send_{j,i}$ is executed after the prefix $\pi$ in $\sigma_P$, then there is a run $\sigma$ common to $P$ and $S_i$ with prefix $\pi send_{j,i}$;

(3) if the action $send_{i,j}$ is executed after the prefix $\pi$ in $\sigma_P$, then there is a run $\sigma$ common to $P$ and $S_i$ with prefix $\pi send'_{i,j}$, with $send'_{i,j}$ possibly different from $send_{i,j}$.

$$\pi$$

(1) $\longrightarrow\longrightarrow\longrightarrow\longrightarrow \quad m(i,j) \longrightarrow\longrightarrow \quad \sigma_S$

$\longrightarrow\longrightarrow\longrightarrow \quad \sigma_P$

$m(i,j) \longrightarrow\longrightarrow \quad \sigma$(of $P$ and $S$)

# Interoperability

**Theorem**

Let $P$ be a protocol with a nonempty set of runs. Let $S_i$ and $S_j$ be two agents that are conformant with $P$.

The interaction of $S$ and $P$ does not produce deadlock situations and it only produces executions of the protocol $P$.

## Reasoning about protocols using automata

Given a DLTL formula $\alpha$ specifying a protocol $P$, we can construct a Büchi automaton $\mathcal{B}_\alpha$ accepting the infinite words corresponding to the runs of $P$.

In particular, we have developed an “on-the-fly” algorithm which extends the one for LTL.

## Verifying the conformance of $S$ with $P$

From the specification of the protocol $P$, we get the nondeterministic automaton $\mathcal{M}_P$.

The behavior of the agent $S$ is given by a *deterministic* Büchi automaton $\mathcal{M}_S$, whose accepted runs provide all the possible executions of the agent.

We define the *synchronous product* between $\mathcal{M}_P$ and $\mathcal{M}_S$

$$\mathcal{M} = \mathcal{M}_P \otimes \mathcal{M}_S$$

whose runs are all the runs of $S$ which are also runs of $P$.
$\mathcal{M}$ is a *non-deterministic generalized* Büchi automaton.

# Verifying the conformance of $S$ with $P$

In order to verify the conformance we must be able to consider all together the states of $\mathcal{M}$ which are reachable with the same prefix.

Unfortunately we know that it is not possible to transform $\mathcal{M}$ into an equivalent deterministic Büchi automaton.
We proceed as follows:

- We define an automaton $\mathcal{M}_{PS}$ by making use of the classical powerset construction for obtaining a deterministic automaton, starting from $\mathcal{M}$.
- For verifying the conformance, we refer to the automaton $\mathcal{M}$, but also make use of the states of the automaton $\mathcal{M}_{PS}$ to reason on the set of states of $\mathcal{M}$ reachable with the same prefix.

## Multiparty protocols

We have extended the approach to protocols involving $k$ agents.

Problem: it is not guaranteed that the constraints in the protocol can be enforced directly by the agents $S_1, \ldots, S_k$.
Consider, for instance, a protocol involving four agents $A, B, C, D$ containing the constraint:

$$[m_1(A, B)] < m_2(C, D) > T$$

meaning that message $m_2$, sent from $C$ to $D$ has to be executed after $m_1$, sent from $A$ to $B$. Assume that $A$ and $B$ do not exchange messages with $C$ and $D$. It is clear that this constraint cannot be enforced by agents $A$ or $B$ alone, as they do not see message $m_2$, nor by agents $C$ or $D$ alone, as they do not see message $m_1$.

## Multiparty protocols

We specify a $k$-role protocol $P$ by specifying separately the $k$ roles.
Given a protocol $P = P_1 \wedge \ldots \wedge P_k$, its runs are obtained by interleaving all the runs of $P_1, \ldots, P_k$, synchronizing on common actions.

We have extended the definition of conformance of an agent $S_i$ with a protocol $P$, by comparing the runs of $S_i$ with those of $P_i$ *in the context* of P.
That is, we define:

$$P[S_i] = P_1 \wedge \ldots \wedge P_{i-1} \wedge S_i \wedge P_{i+1} \wedge \ldots \wedge P_k$$

and we compare the runs of $P[S_i]$ with those of $P$.