MASSiVE, Unità di Torino

Verifying the Conformance of Web Services to Global Interaction Protocols

M. Baldoni, C. Baroglio, A. Martelli, V. Patti, C. Schifanella
Dipartimento di Informatica – Univ. degli Studi di Torino
C.so Svizzera, 185, I-10149 Torino (Italy)
http://www.di.unito.it/~argo
{baldoni,baroglio,mrt,patti,schi}@di.unito.it

A network of web services

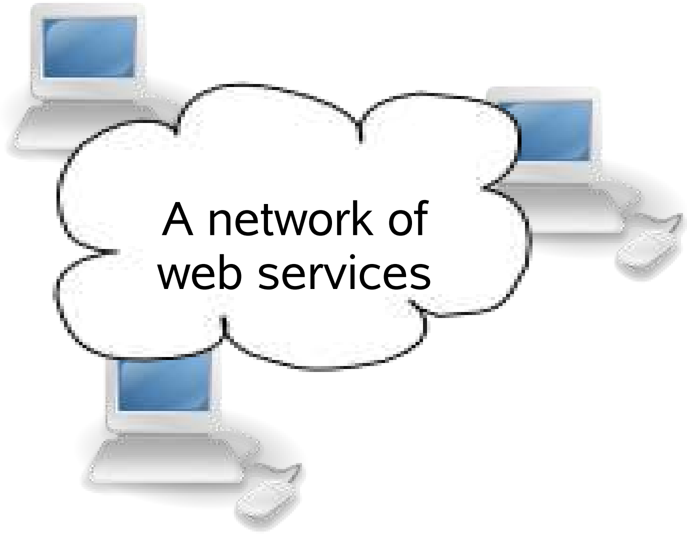
- Web services are heterogeneous devices
- Executable description of their business process (expecially the interactive behaviour)
- Composition

uses

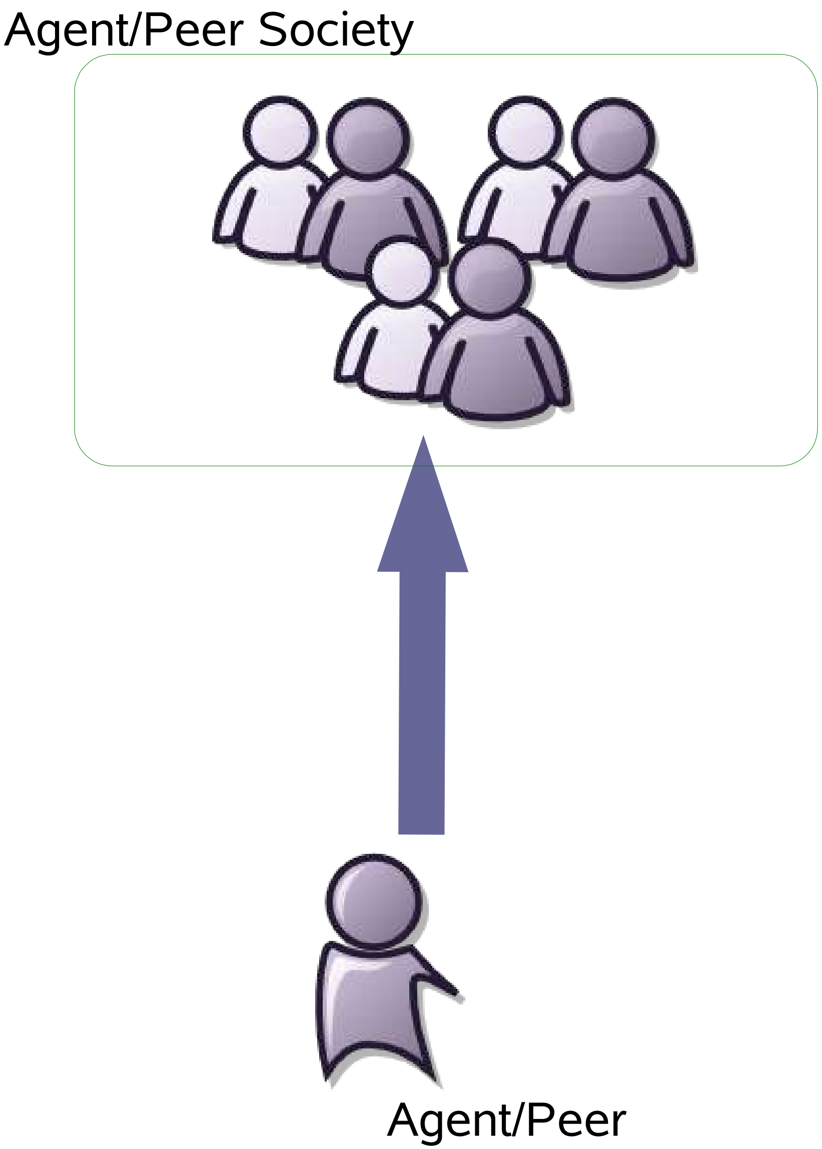
- Selection

uses

- Web services share some similarities with agents
- Interoperability problem



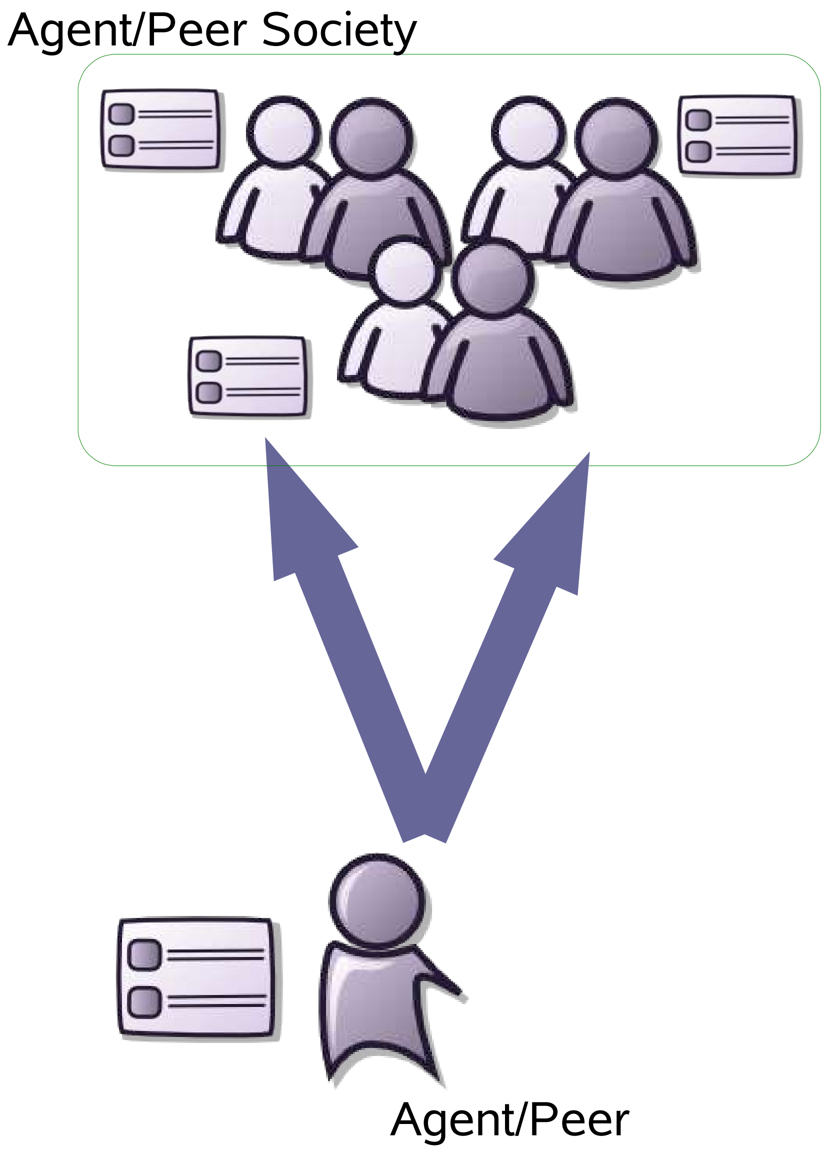
Interoperability



- *“Will the agent/peer able to produce a conversation with the members of the group?”*
- Interoperability is the capability of an agent/peer of interacting with others
- This means it will actually produce a “complete” conversation with them

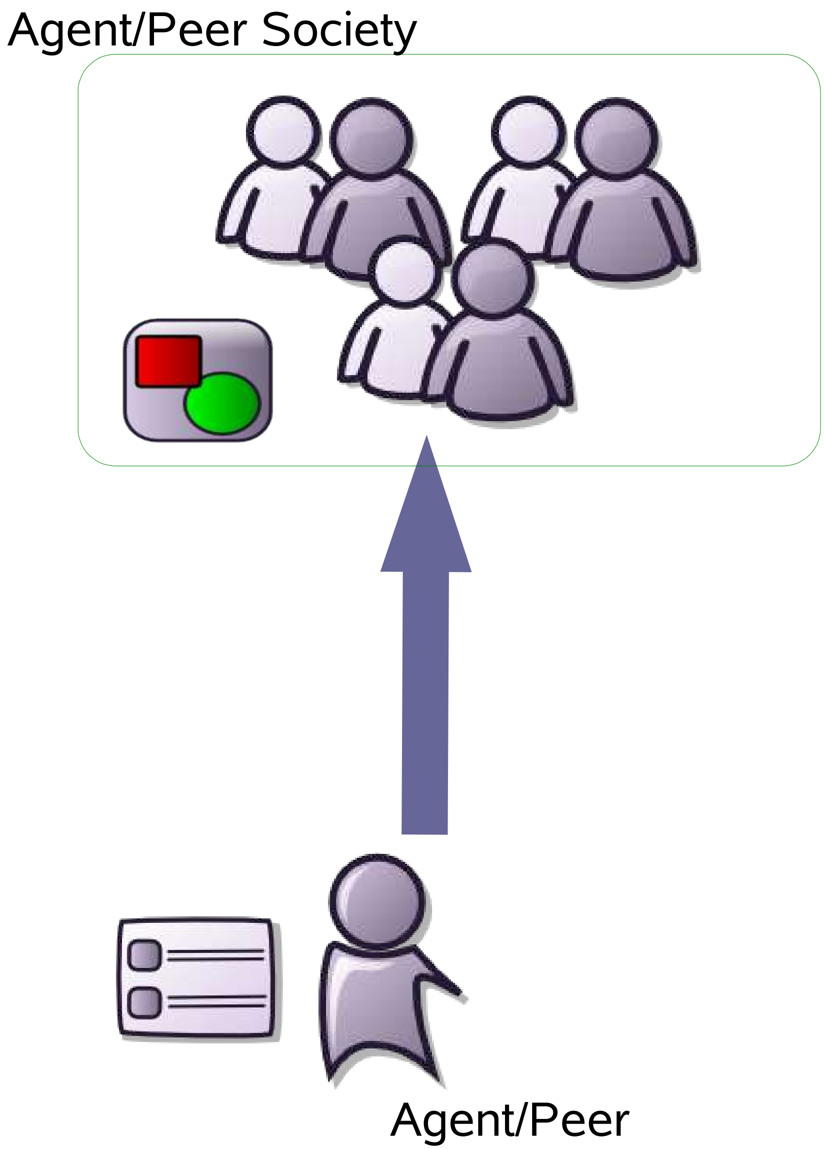
This means that they will not interrupt their conversation

Checking interoperability



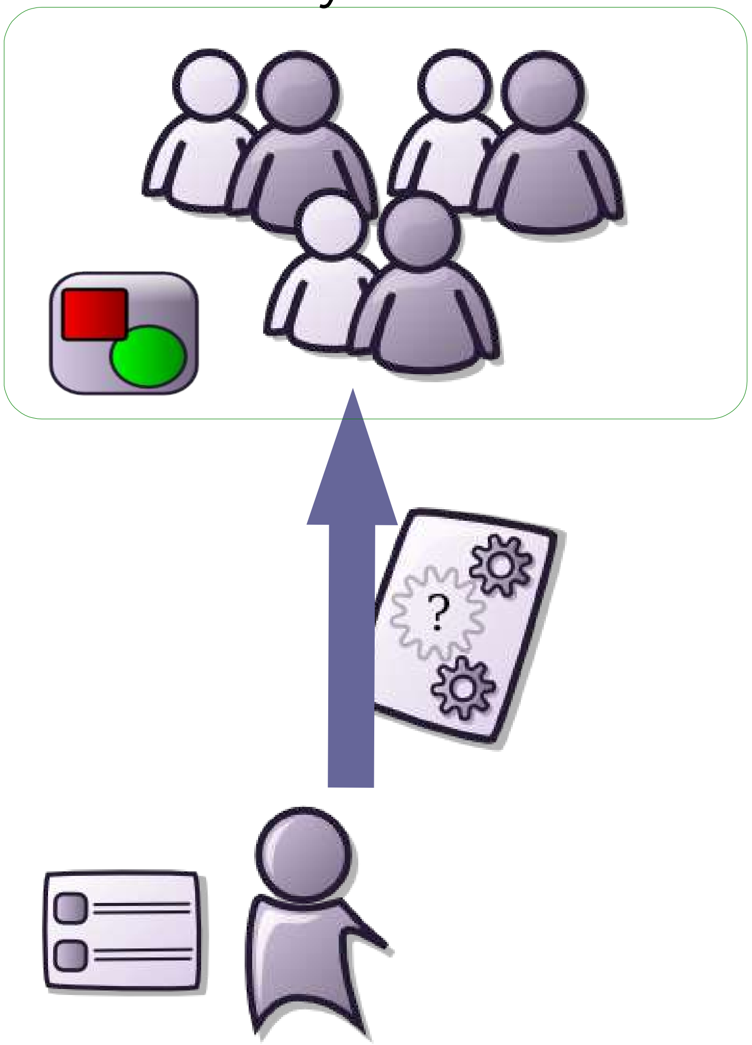
- Either we verify the interaction of each agent/peer with each other

Checking interoperability



- Either we verify the interaction of each agent/peer with each other
- Or we introduce a set of rules that determine the overall behavior: an interaction protocol

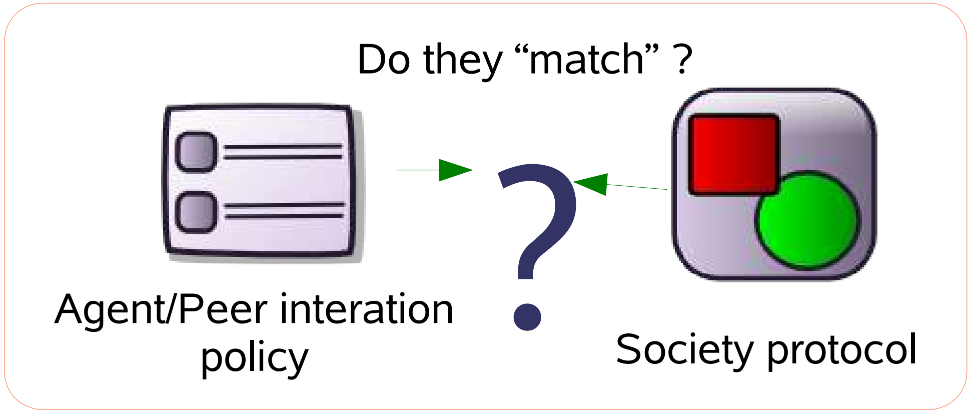
Checking interoperability



Agent/Peer Society

Agent/Peer

- Either we verify the interaction of each agent with each other
- Or we introduce a set of rules that determine the overall behavior: an interaction protocol
- Agent/peer policy against society protocol

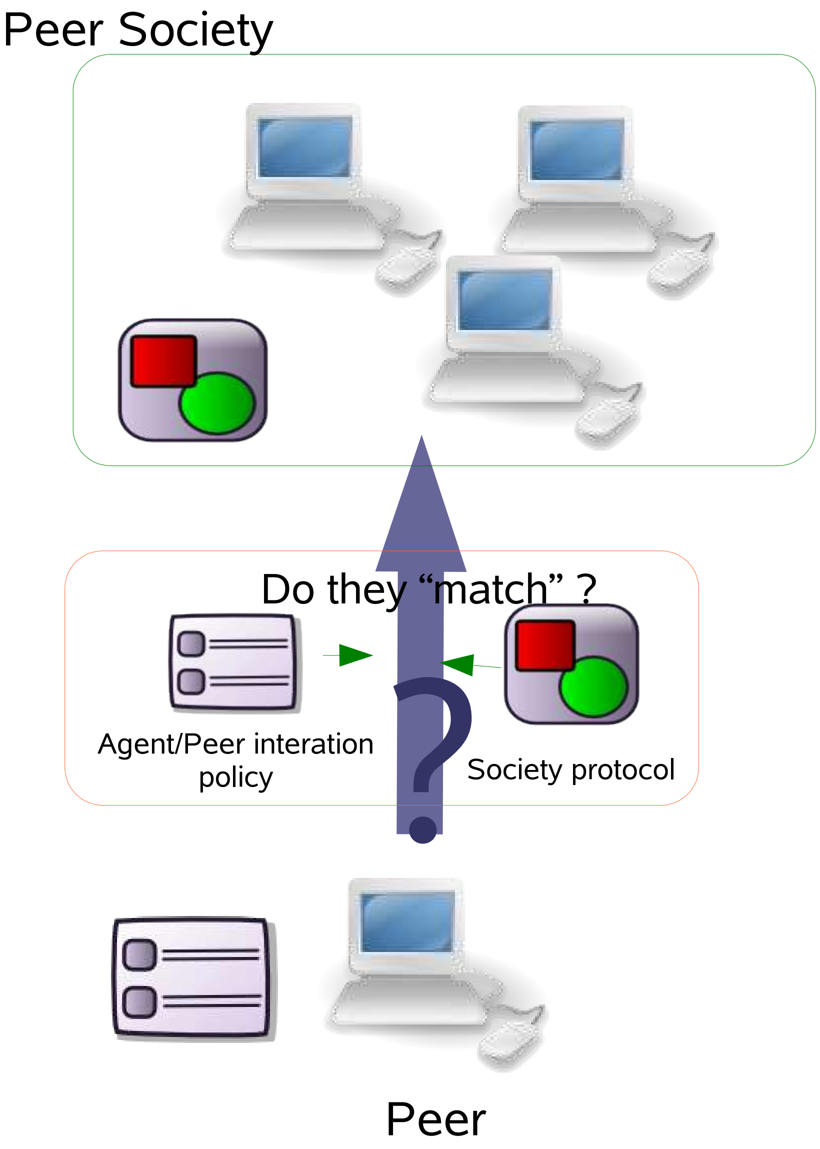


Do they “match” ?

Agent/Peer interation policy

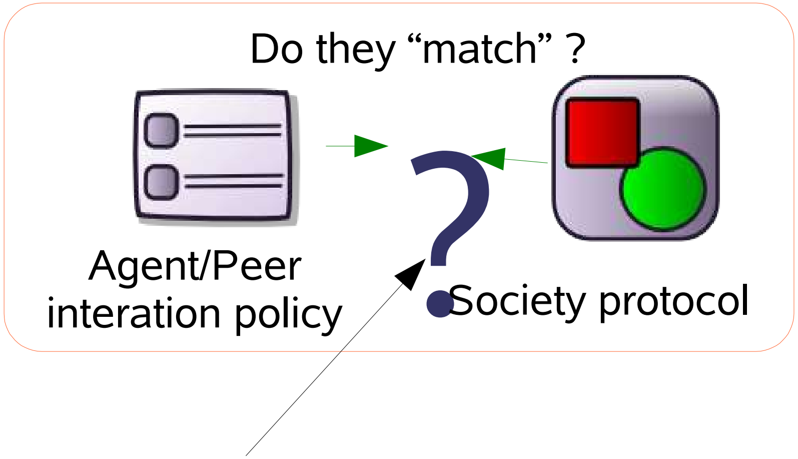
Society protocol

Checking interoperability: web services



- Choreography: global point of view/abstract protocol, eg. WS-CDL language
- Behavioral interface: local point of view/policy, eg. BPEL abstract process
- Orchestration: describes both communicative and non-coomunicative behaviour allowing execution, eg. BPEL executable process

Conformance test should entail interoperability



Do they “match” ?

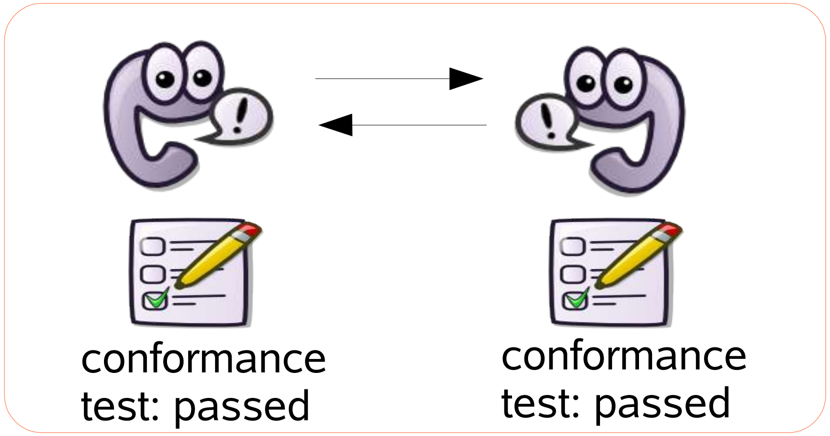
Agent/Peer
interation policy

Society protocol

Verifying if a given implementation (policy) respects an abstract protocol definition is known as (a priori) conformance test

For logic agents: [Endriss, Maudet, Sadri, Toni; 2003, 2004]

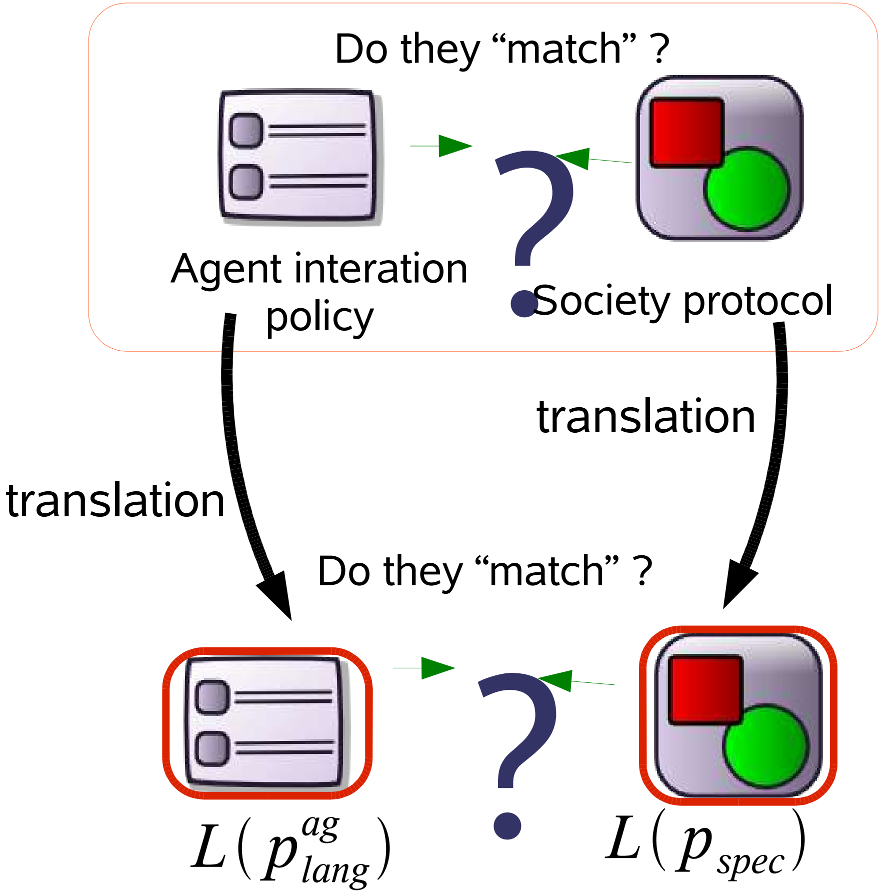
- A conformance test should be proved to imply interoperability
- We expect that two agents, that individually conform to a protocol, will actually produce a legal and “complete” conversation



conformance
test: passed

conformance
test: passed

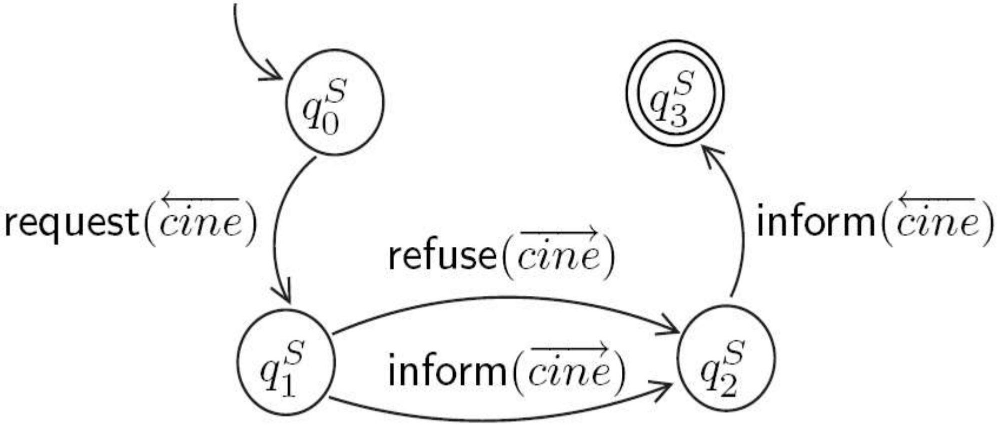
A conformance test [CLIMA VI]



- We define an a-priori conformance test that guarantees interoperability
- Based on formal languages: protocols and policies represented as regular languages
- Conformance test: acceptance of both languages by a special finite state automaton

Protocols as regualar languages

request($\overleftarrow{\text{cine}}$) ≡ request(customer, cine, movie)
inform($\overrightarrow{\text{cine}}$) ≡ inform(cine, customer, movie)

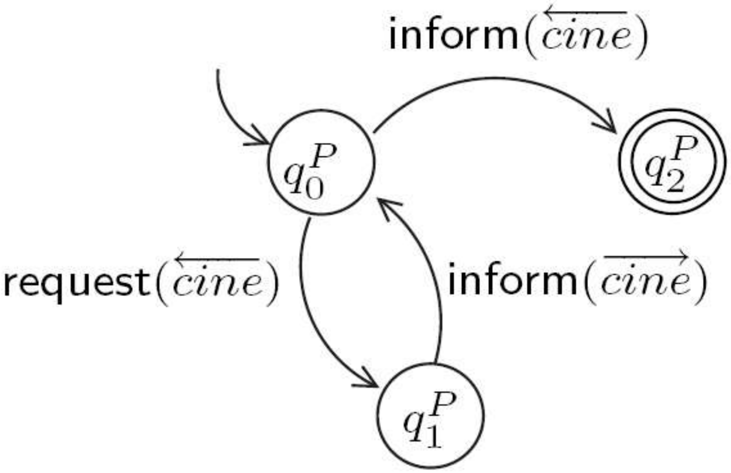

q_0^S
q_3^S
request($\overleftarrow{cine}$)
inform($\overleftarrow{cine}$)
refuse($\overrightarrow{cine}$)
q_1^S
inform($\overrightarrow{cine}$)
q_2^S

The protocol adopted by a cinema web service: it waits for a request about a movie, it sends a feedback (give or refuse an information) and waits for an acknowledgement

- A protocol p_{spec} is represented as an FSM
- Speech acts of the form $m(ag_s, ag_r, l)$
- Conversations are sequences of speech acts
- $L(p_{spec})$ is the set of all the conversations allowed by the protocol p_{spec}, that is accepted by its automaton

Policies as regualar languages

request($\overleftarrow{\text{cine}}$) ≡ request(customer, cine, movie)
inform($\overrightarrow{\text{cine}}$) ≡ inform(cine, customer, movie)

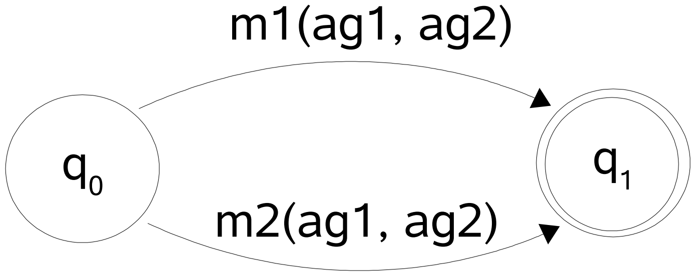


inform($\overleftarrow{cine}$)
q_0^P
q_2^P
request($\overleftarrow{cine}$)
inform($\overrightarrow{cine}$)
q_1^P

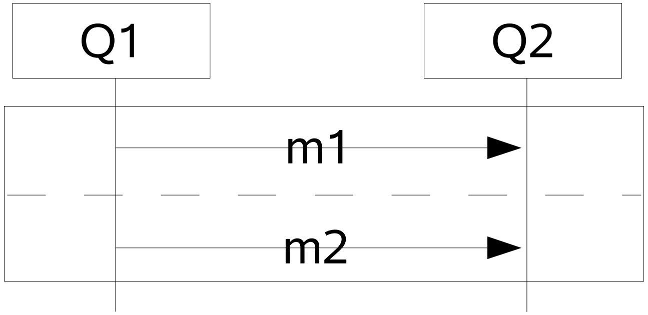
The policy of a specific cinema web service with a reactive behavior: on request informs, while on acknowledgement it stops

- A policy p_{lang} is represented as an FSM
- Speech acts of the form $m(ag_s, ag_r, l)$
- Conversations are sequences of speech acts
- $L(p_{lang})$ is the set of all the conversations allowed by the policy p_{lang}, that is accepted by its automaton

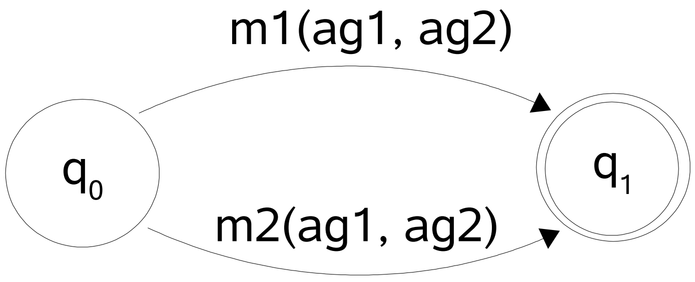
Conformance?

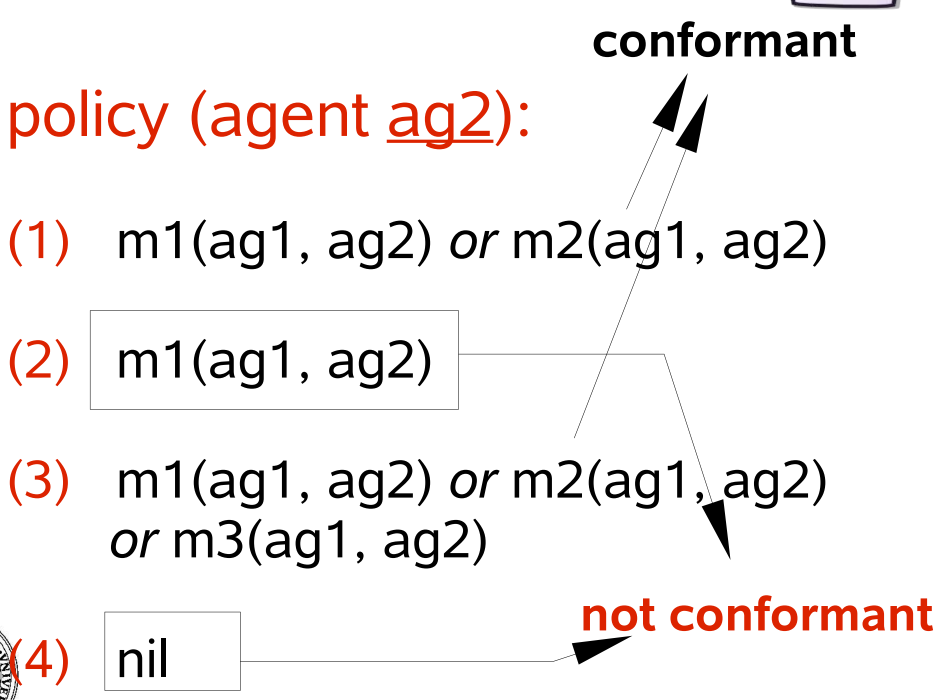


- A simple protocol: an agent can send to another agent either the message m1 or the message m2



Conformance?





conformant

policy (agent ag2):

(1) m1(ag1, ag2) *or* m2(ag1, ag2)

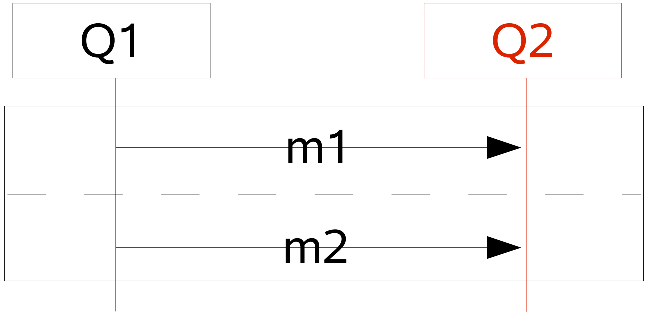
(2) m1(ag1, ag2)

(3) m1(ag1, ag2) *or* m2(ag1, ag2) *or* m3(ag1, ag2)

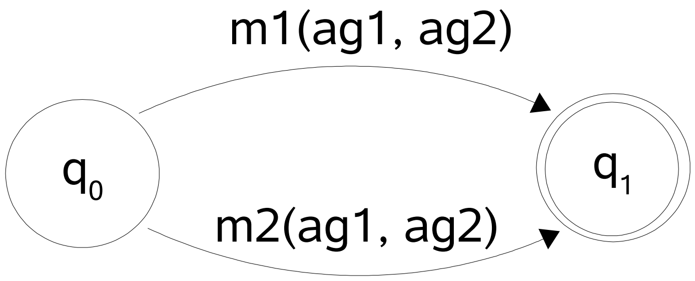
(4) nil

not conformant

- A policy must handle any incoming message, foreseen by the protocol
- The capability of handling further incoming messages does not compromise conformance (such messages will never be received when dealing with a conformant partner)



Conformance?



conformant

policy (agent ag1):

(1) m1(ag1, ag2) *or* m2(ag1, ag2)

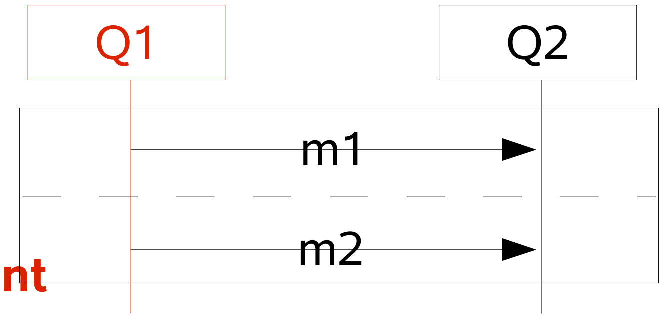
(2) m1(ag1, ag2)

(3) m1(ag1, ag2) *or* m2(ag1, ag2) *or* m3(ag1, ag2)

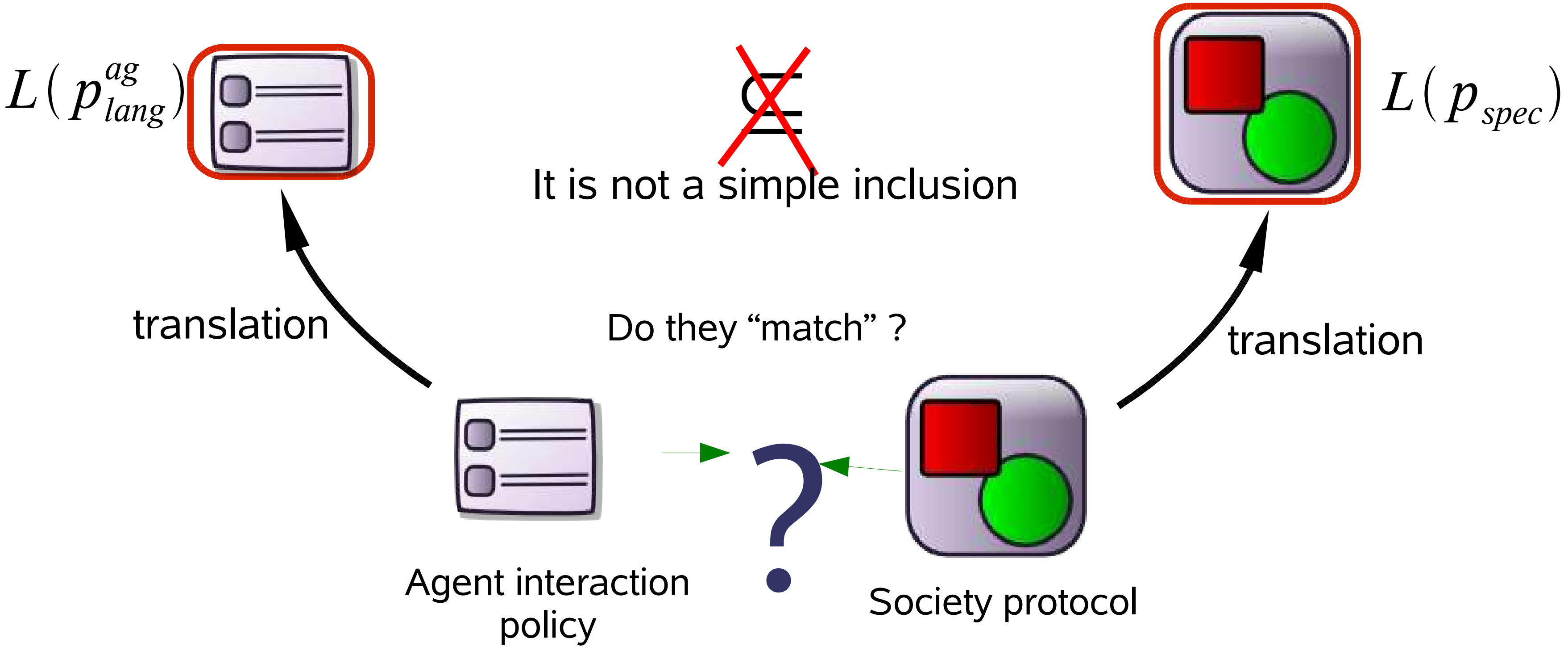
(4) nil

not conformant

- It must not foresee any outgoing message that is not foreseen by the protocol
- A policy must allow uttering at least one of the outgoing messages, foreseen by the protocol
- A policy does not have to foresee all of the alternative outgoing messages



Conformance?



$L(p^{ag}_{lang})$

$\xcancel{\subseteq}$

It is not a simple inclusion

$L(p_{spec})$

translation

Do they “match” ?

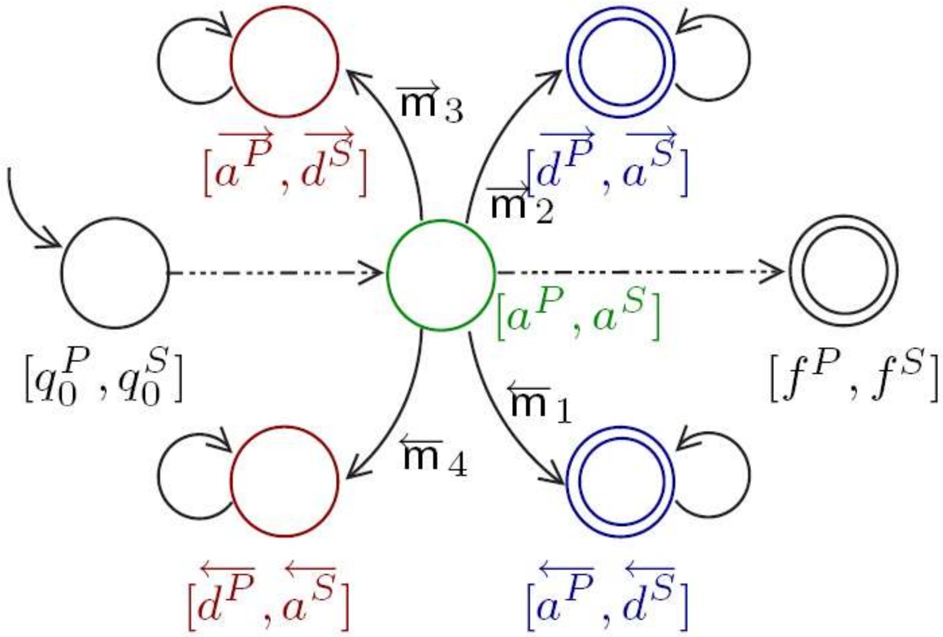
translation

Agent interaction policy

Society protocol

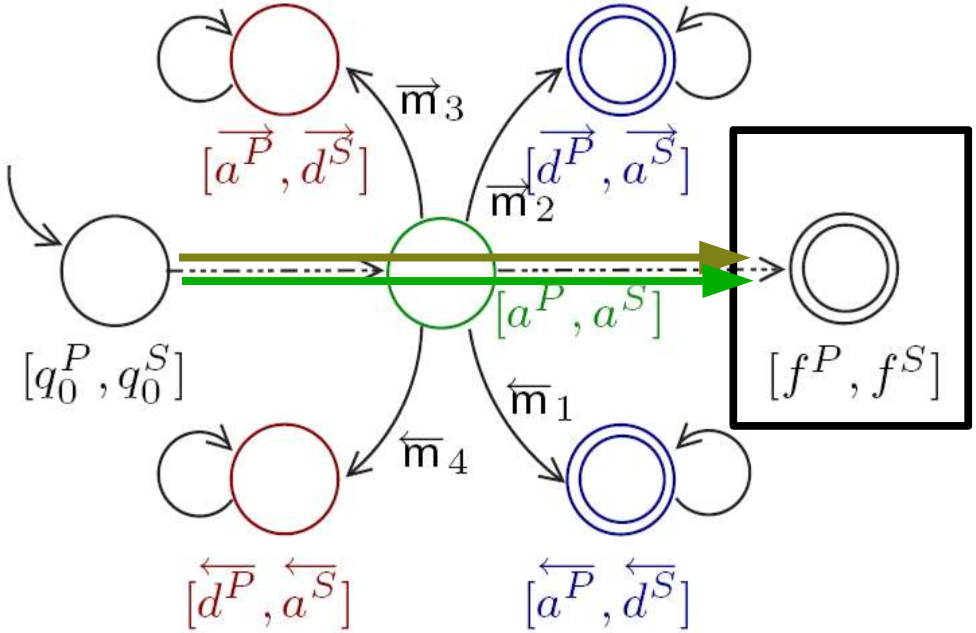
At every point of conversation, we expect that a conformant policy never utters speech acts that are not expected, according to the protocol, and we also expect it to be able to handle any message that can possibly be received, according to the protocol. however, the policy is not obliged to foresee an ougoing message for every alternative included in the protocol (but at least one of them)

The automaton M_{conf}


$[\overrightarrow{a}^P, \overrightarrow{d}^S]$
$\overrightarrow{\mathsf{m}}_3$
$[\overrightarrow{d}^P, \overrightarrow{a}^S]$
$\overrightarrow{\mathsf{m}}_2$
$[q_0^P, q_0^S]$
$[a^P, a^S]$
$[f^P, f^S]$
$\overleftarrow{\mathsf{m}}_1$
$\overleftarrow{\mathsf{m}}_4$
$[\overleftarrow{d}^P, \overleftarrow{a}^S]$
$[\overleftarrow{a}^P, \overleftarrow{d}^S]$

- Verifying that both languages *L(p*$_{spec}$*)* and *L(p*$_{lang}$*)* are recognized by a special finite state automaton, called *M*$_{conf}$
- *M*$_{conf}$ is based on the automaton that recognizes the intersection of *L(p*$_{spec}$*)* and *L(p*$_{lang}$*)* ...
- ... and it captures the expectations about conformance introduced earlier: some more conversations are to be accepted, some surely not

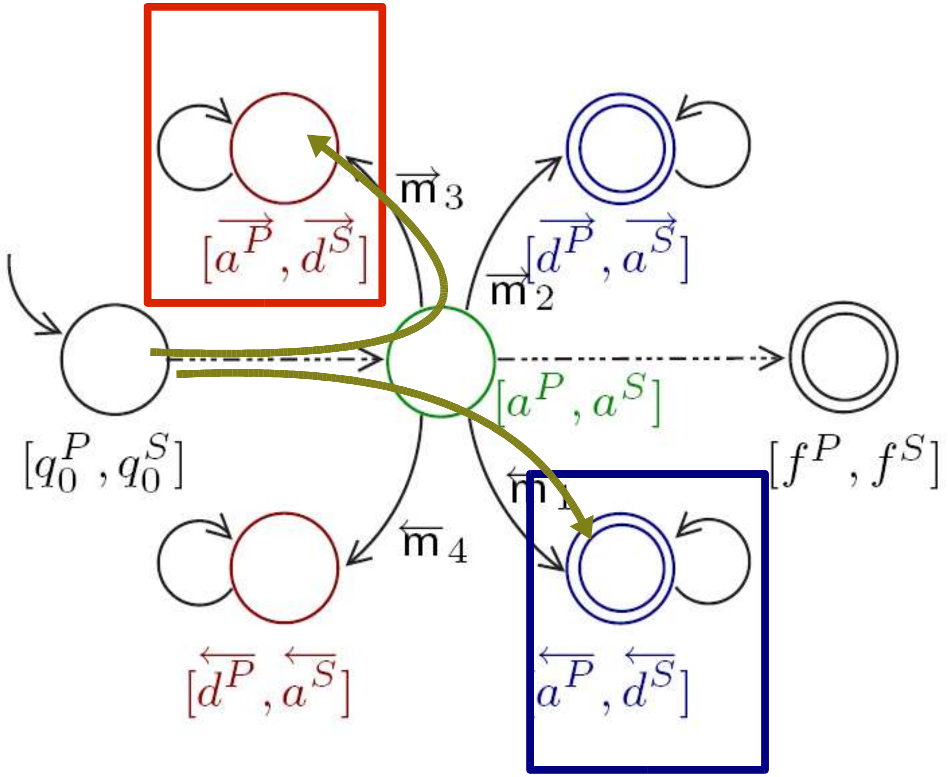
The automaton M_{conf}


$[\overrightarrow{a^P}, \overrightarrow{d^S}]$
$\overrightarrow{m}_3$
$[\overrightarrow{d^P}, \overrightarrow{a^S}]$
$\overrightarrow{m}_2$
$[q_0^P, q_0^S]$
$[a^P, a^S]$
$[f^P, f^S]$
$\overleftarrow{m}_1$
$\overleftarrow{m}_4$
$[\overleftarrow{d^P}, \overleftarrow{a^S}]$
$[\overleftarrow{a^P}, \overleftarrow{d^S}]$

- The conversations that belong to the intersection of the two languages are surely legal and we wish to accept them

The automaton M_{conf}

not accepted



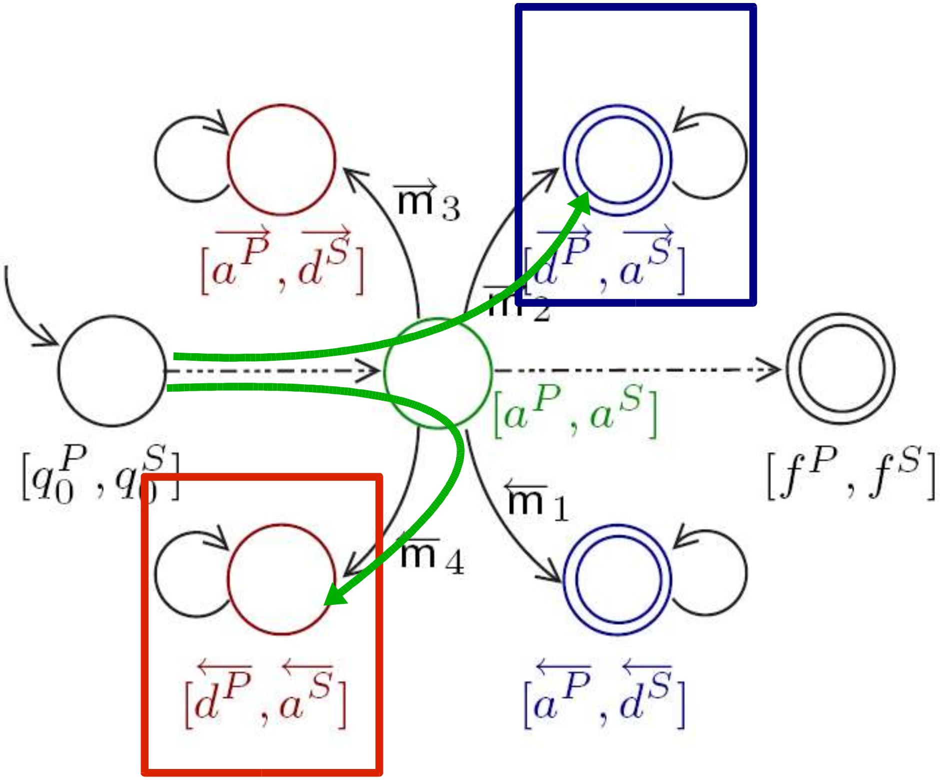
$[\overrightarrow{a}^P, \overrightarrow{d}^S]$
$\overrightarrow{m}_3$
$[\overrightarrow{d}^P, \overrightarrow{a}^S]$
$\overrightarrow{m}_2$
$[q_0^P, q_0^S]$
$[a^P, a^S]$
$[f^P, f^S]$
$\overleftarrow{m}_1$
$\overleftarrow{m}_4$
$[\overleftarrow{d}^P, \overleftarrow{a}^S]$
$[\overleftarrow{a}^P, \overleftarrow{d}^S]$

accepted

- For conversations in the policy that do not belong to the specification
- we want to accept the ones which, at a certain point, according to the policy, allow the reception of a message that, instead, cannot be received according the specification
- we do not want to accept the ones which at a certain point, according to the policy, allow the utterance of a message, that is not possible according the specification

The automaton M_{conf}

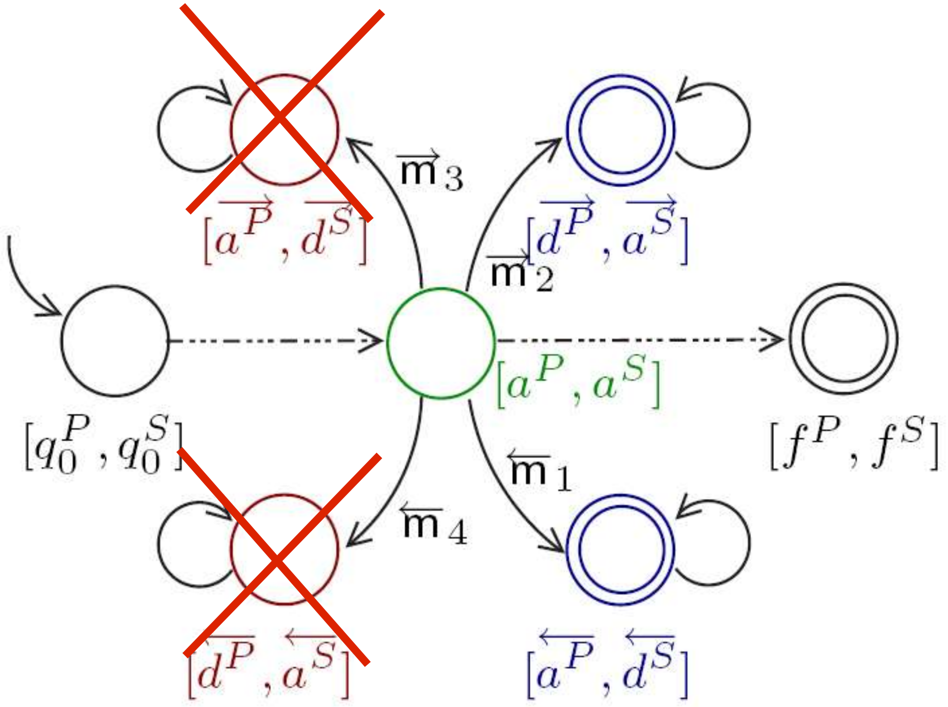
accepted


$\overrightarrow{m}_3$
$[\overrightarrow{a}^P, \overrightarrow{d}^S]$
$[\overrightarrow{d}^P, \overrightarrow{a}^S]$
$\overrightarrow{m}_2$
$[q_0^P, q_0^S]$
$[a^P, a^S]$
$[f^P, f^S]$
$\overleftarrow{m}_1$
$\overleftarrow{m}_4$
$[\overleftarrow{d}^P, \overleftarrow{a}^S]$
$[\overleftarrow{a}^P, \overleftarrow{d}^S]$

not accepted

- For conversations in the specification that do not belong to the policy
- we want to accept the ones which, at a certain point, according to the specification,allow the utterance of a message that, instead, the agent will never utter
- we do not want to accept the ones which, at a certain point, according to the specification,allow the reception of a message, that is not dealt with by the agent

The automaton M_{conf}



$[\overrightarrow{a^P}, \overrightarrow{d^S}]$
$\overrightarrow{m}_3$
$[\overrightarrow{d^P}, \overrightarrow{a^S}]$
$\overrightarrow{m}_2$
$[q_0^P, q_0^S]$
$[a^P, a^S]$
$[f^P, f^S]$
$\overleftarrow{m}_1$
$\overleftarrow{m}_4$
$[\overleftarrow{d^P}, \overleftarrow{a^S}]$
$[\overleftarrow{a^P}, \overleftarrow{d^S}]$

- Actually, we expect no conversation will lead to red states

- Complete automaton:
- Whenever, w.r.t. the specification, the agent is supposed to utter a message (out of some alternatives), its policy allows at least one of such alternatives

Policy conformance test

Definition A policy p_{lang} is conformant to a protocol specification p_{spec} iff the automaton M_{conf} is complete and it accepts both languages $L(p_{lang})$ and $L(p_{spec})$.

Proposition Given a policy p_{lang} that is conformant to a protocol specification p_{spec} for every prefix σ' that is common to the two languages $L(p_{lang})$ and $L(p_{spec})$, there is a conversation $\sigma = \sigma'\sigma''$ such that σ is in the intersection of $L(p_{lang})$ and $L(p_{spec})$.

If the automaton is not complete we cannot guarantee that the agent will be able to conclude its conversation, but only that its conversations, if any, will be legal

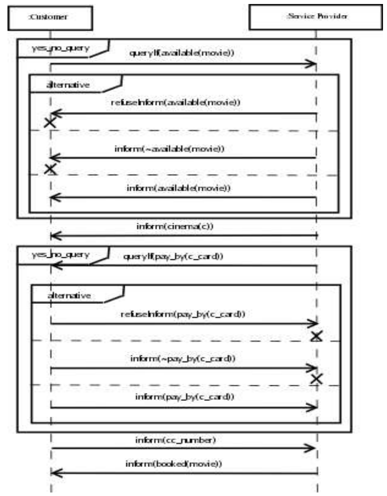
Interoperability and decidability

... Then the intersection of the two languages cannot be empty (it contains at least one legal conversation) and the two languages do not necessarily coincide

<u>Theorem 1</u> (*Interoperability*). For every prefix σ' that is common to the two languages $L(p_{lang1})$ and $L(p_{lang2})$, there is a conversation $\sigma = \sigma'\sigma''$ such that σ belongs to the intersection between $L(p_{lang1})$ and $L(p_{lang2})$.

<u>Theorem 2</u> (*Decidability*). Policy conformance is decidable when $L(p_{lang})$ and $L(p_{spec})$ are regualar languages.

Translation: an example [CLIMA V]



AUML
interaction
diagram

extract

Formal Language:
it represents all
possible sequences
of dialogue acts on the
basis of the AUML
sequence diagram

Conformance
test

Different sets of possible dialogues
depending on the level of abstraction from
the agent mental state

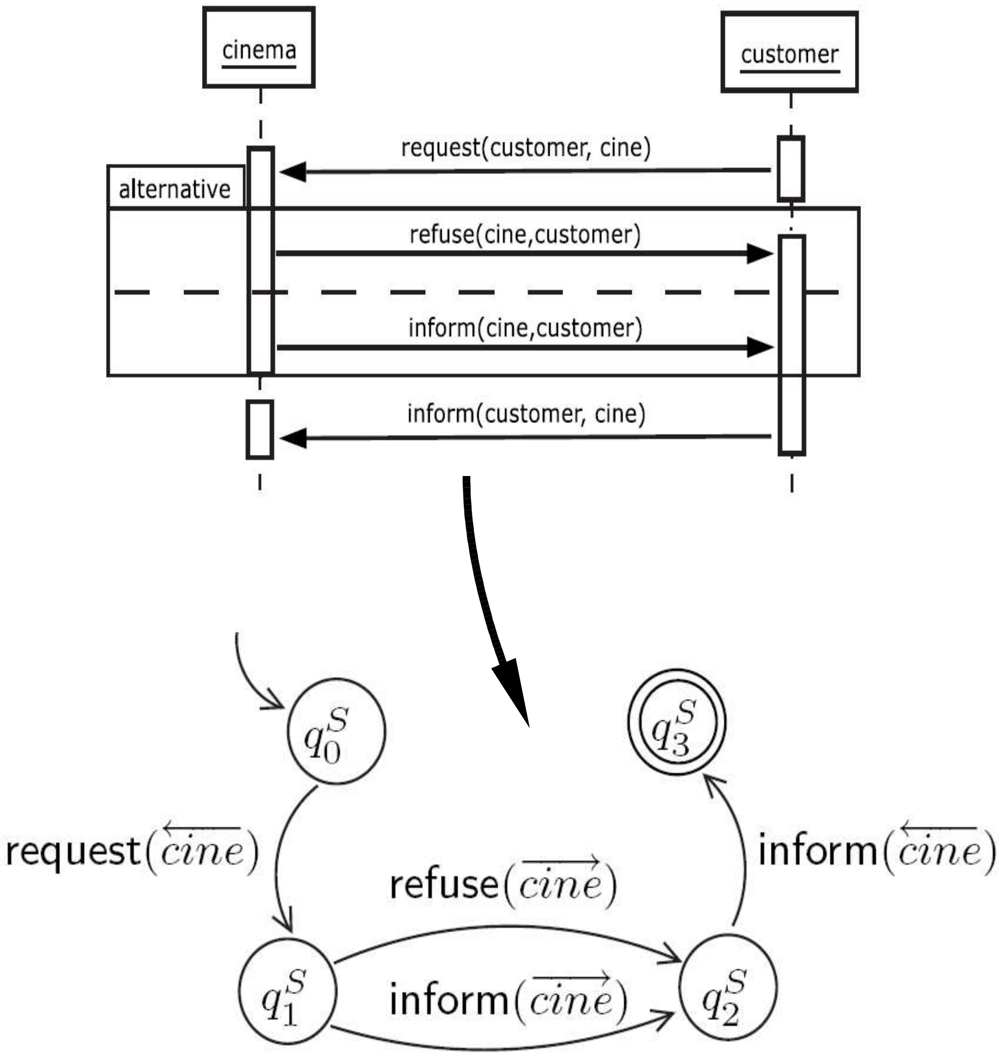
Sequences corresponding to all possible dialogues allowed by the implementation

extract

DyLOG
implementation

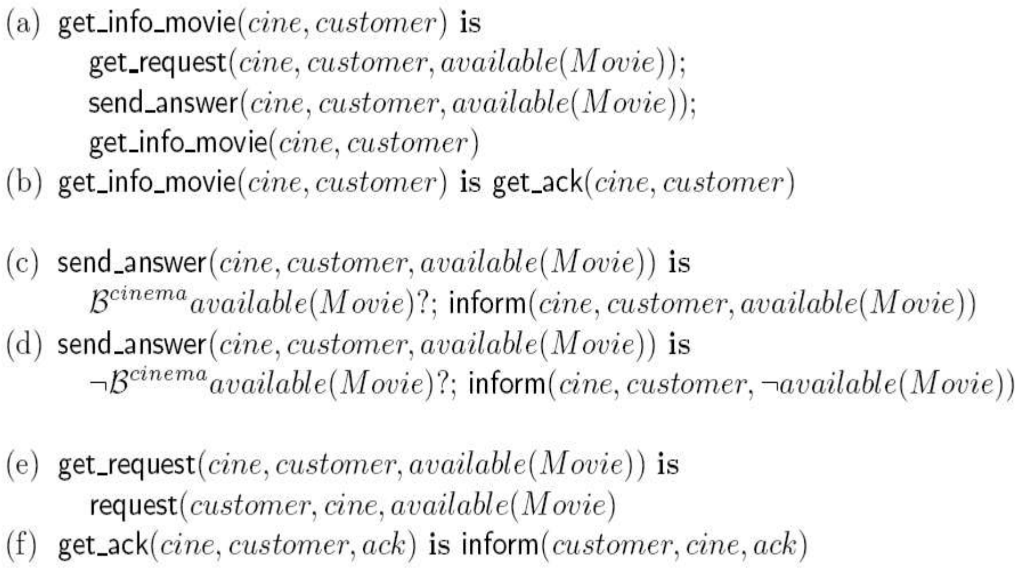
$$
\begin{aligned}
&\langle \mathsf{reserv\_rest\_1}_C(Self, Service, Time)\rangle\varphi \subset \\
&\quad \langle \mathsf{yes\_no\_query}_Q(Self, Service, available(Time))\ ; \\
&\qquad \mathcal{B}^{Self} available(Time)?\ ; \\
&\qquad \mathsf{get\_info}(Self, Service, reservation(Time))\ ; \\
&\qquad \mathsf{get\_info}(Self, Service, cinema\_promo)\ ; \\
&\qquad \mathsf{get\_info}(Self, Service, ft\_number)\rangle\varphi
\end{aligned}
$$

Protocol translation: AUML to FSM



- This translation can be done by following the algorithm 1 described in CLIMA V

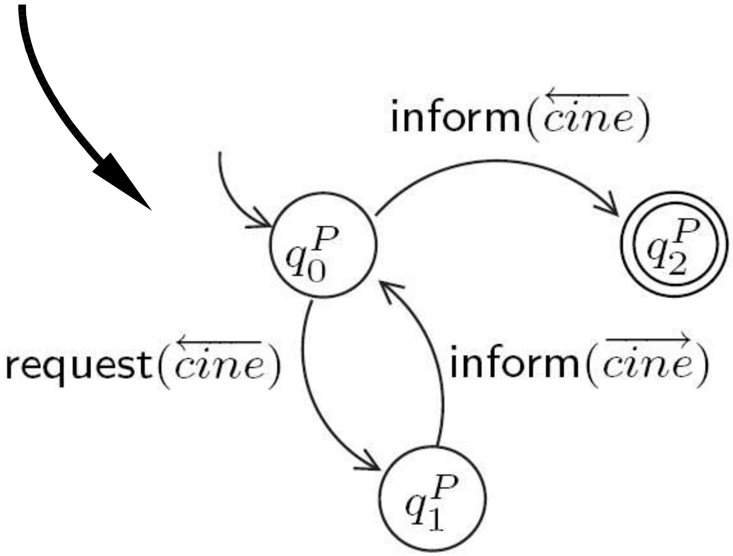
Policy translation: DyLOG to FSM



(a) get_info_movie$(cine, customer)$ is
  get_request$(cine, customer, available(Movie))$;
  send_answer$(cine, customer, available(Movie))$;
  get_info_movie$(cine, customer)$
(b) get_info_movie$(cine, customer)$ is get_ack$(cine, customer)$

(c) send_answer$(cine, customer, available(Movie))$ is
  $\mathcal{B}^{cinema} available(Movie)?$; inform$(cine, customer, available(Movie))$
(d) send_answer$(cine, customer, available(Movie))$ is
  $\neg\mathcal{B}^{cinema} available(Movie)?$; inform$(cine, customer, \neg available(Movie))$

(e) get_request$(cine, customer, available(Movie))$ is
  request$(customer, cine, available(Movie)$
(f) get_ack$(cine, customer, ack)$ is inform$(customer, cine, ack)$


inform($\overleftarrow{cine}$)
q_0^P
q_2^P
request($\overleftarrow{cine}$)
inform($\overrightarrow{cine}$)
q_1^P

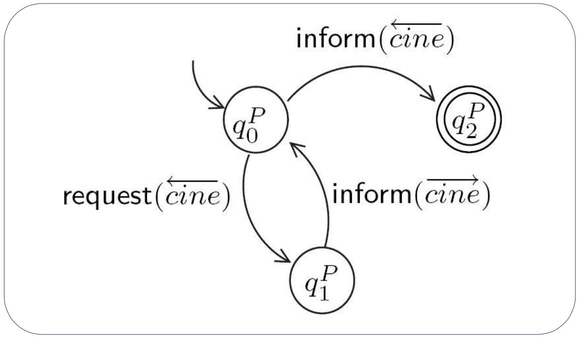
- This can be done by algorithm 2 of CLIMA V
- It exploits the form of inclusion axioms used to encode conversation policies:

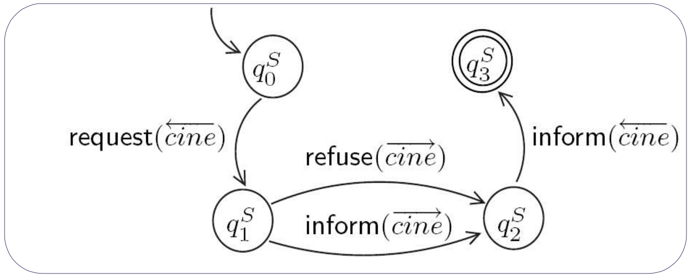
$$\langle p_0 \rangle \varphi \subset \langle p_1 \rangle \langle p_2 \rangle \cdots \langle p_n \rangle \varphi$$

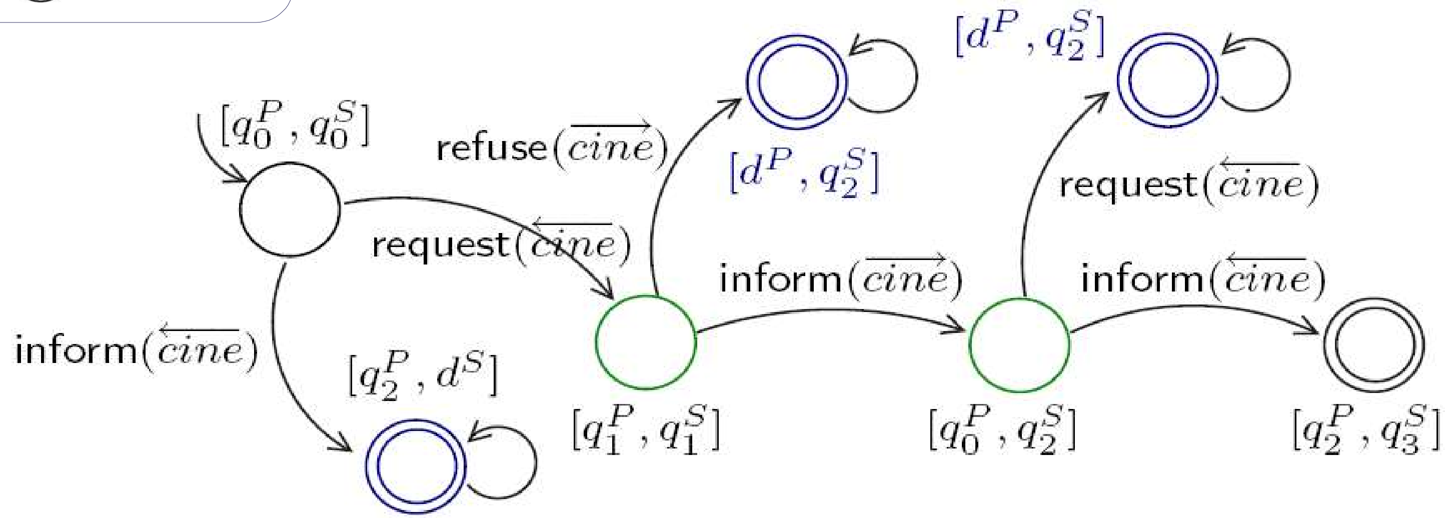
$$\Big\downarrow$$

$$p_0 \rightarrow p_1\, p_2 \cdots p_n$$

The automaton M_{conf}

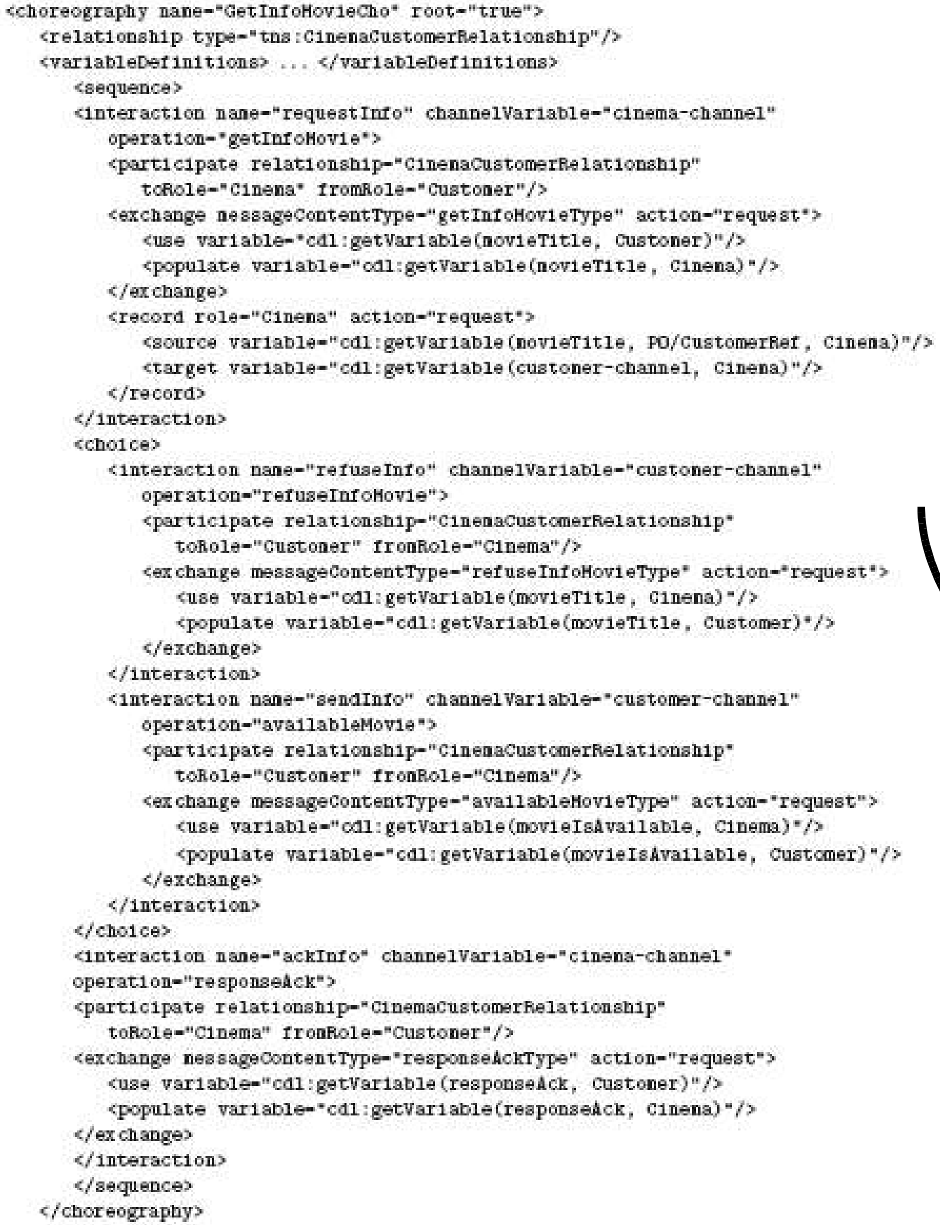

inform($\overleftarrow{cine}$)
q_0^P
q_2^P
request($\overleftarrow{cine}$)
inform($\overrightarrow{cine}$)
q_1^P


q_0^S
q_3^S
request($\overleftarrow{cine}$)
refuse($\overrightarrow{cine}$)
inform($\overleftarrow{cine}$)
q_1^S
inform($\overrightarrow{cine}$)
q_2^S


$[d^P, q_2^S]$
$[q_0^P, q_0^S]$
refuse($\overrightarrow{cine}$)
$[d^P, q_2^S]$
request($\overleftarrow{cine}$)
request($\overleftarrow{cine}$)
inform($\overrightarrow{cine}$)
inform($\overleftarrow{cine}$)
inform($\overleftarrow{cine}$)
$[q_2^P, d^S]$
$[q_1^P, q_1^S]$
$[q_0^P, q_2^S]$
$[q_2^P, q_3^S]$

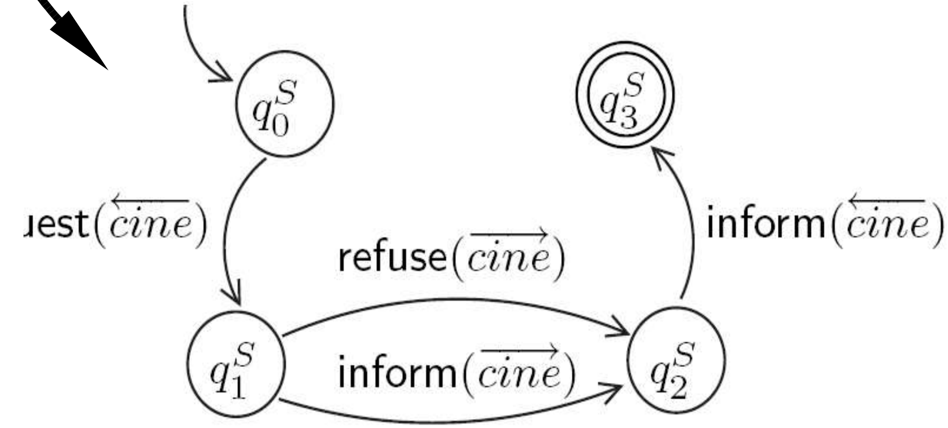
- The automaton is complete and accepts both languages
- The agent's policy is conformant and interoperable

Protocol translation: WS-CDL to FSM



```
<choreography name="GetInfoMovieCho" root="true">
  <relationship type="tns:CinemaCustomerRelationship"/>
  <variableDefinitions> ... </variableDefinitions>
    <sequence>
    <interaction name="requestInfo" channelVariable="cinema-channel"
      operation="getInfoMovie">
      <participate relationship="CinemaCustomerRelationship"
        toRole="Cinema" fromRole="Customer"/>
      <exchange messageContentType="getInfoMovieType" action="request">
        <use variable="cdl:getVariable(movieTitle, Customer)"/>
        <populate variable="cdl:getVariable(movieTitle, Cinema)"/>
      </exchange>
      <record role="Cinema" action="request">
        <source variable="cdl:getVariable(movieTitle, PO/CustomerRef, Cinema)"/>
        <target variable="cdl:getVariable(customer-channel, Cinema)"/>
      </record>
    </interaction>
    <choice>
      <interaction name="refuseInfo" channelVariable="customer-channel"
        operation="refuseInfoMovie">
        <participate relationship="CinemaCustomerRelationship"
          toRole="Customer" fromRole="Cinema"/>
        <exchange messageContentType="refuseInfoMovieType" action="request">
          <use variable="cdl:getVariable(movieTitle, Cinema)"/>
          <populate variable="cdl:getVariable(movieTitle, Customer)"/>
        </exchange>
      </interaction>
      <interaction name="sendInfo" channelVariable="customer-channel"
        operation="availableMovie">
        <participate relationship="CinemaCustomerRelationship"
          toRole="Customer" fromRole="Cinema"/>
        <exchange messageContentType="availableMovieType" action="request">
          <use variable="cdl:getVariable(movieIsAvailable, Cinema)"/>
          <populate variable="cdl:getVariable(movieIsAvailable, Customer)"/>
        </exchange>
      </interaction>
    </choice>
    <interaction name="ackInfo" channelVariable="cinema-channel"
    operation="responseAck">
    <participate relationship="CinemaCustomerRelationship"
      toRole="Cinema" fromRole="Customer"/>
    <exchange messageContentType="responseAckType" action="request">
      <use variable="cdl:getVariable(responseAck, Customer)"/>
      <populate variable="cdl:getVariable(responseAck, Cinema)"/>
    </exchange>
    </interaction>
    </sequence>
  </choreography>
```

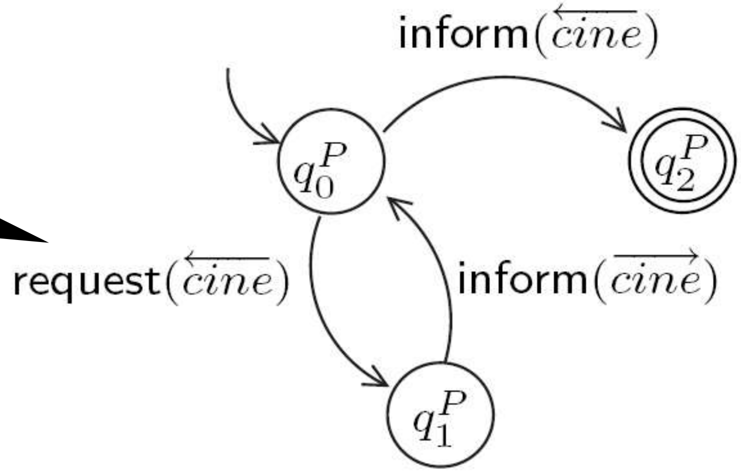
Translating WS-CDL to FSM


q_0^S
q_3^S
uest($\overleftarrow{cine}$)
inform($\overleftarrow{cine}$)
refuse($\overrightarrow{cine}$)
q_1^S
inform($\overrightarrow{cine}$)
q_2^S

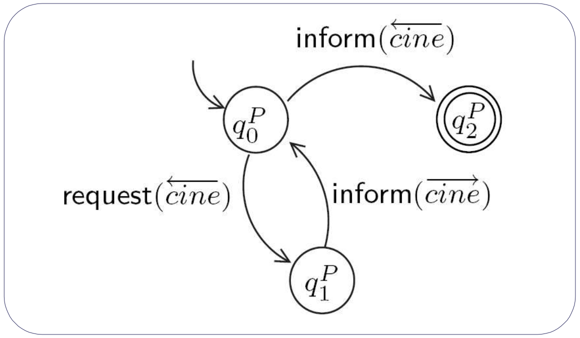
Policy translation: BPEL4WS to FSM

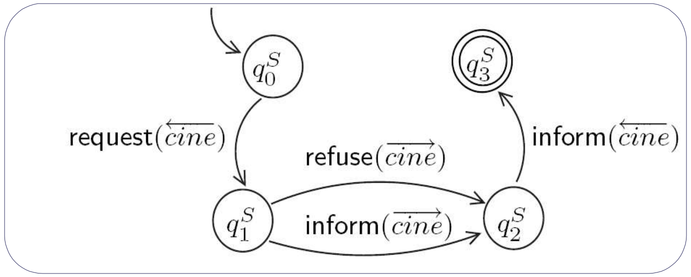
```
<sequence>
   ...
   <while condition="bpws:getVariableData('done') = 'false'">
      <pick>
         <onMessage portType="movieInfoPT" partnerLink="customer"
            operation="movieInfoACK">
            <assign>
               <copy>
                  <from expression="true" />
                  <to variable="done" />
               </copy>
            </assign>
         </onMessage>
         <onMessage portType="movieInfoPT" partnerLink="customer"
            operation="movieInfo" variable="movieTitle">
            <sequence>
               ... retrieve information ...
               <reply portType="customerPT" partnerLink="customer"
                  operation="informMovieAvailable"
                  variable="movieAvailable">
               </reply>
            </sequence>
         </onMessage>
      </pick>
   </while>
</sequence>
```

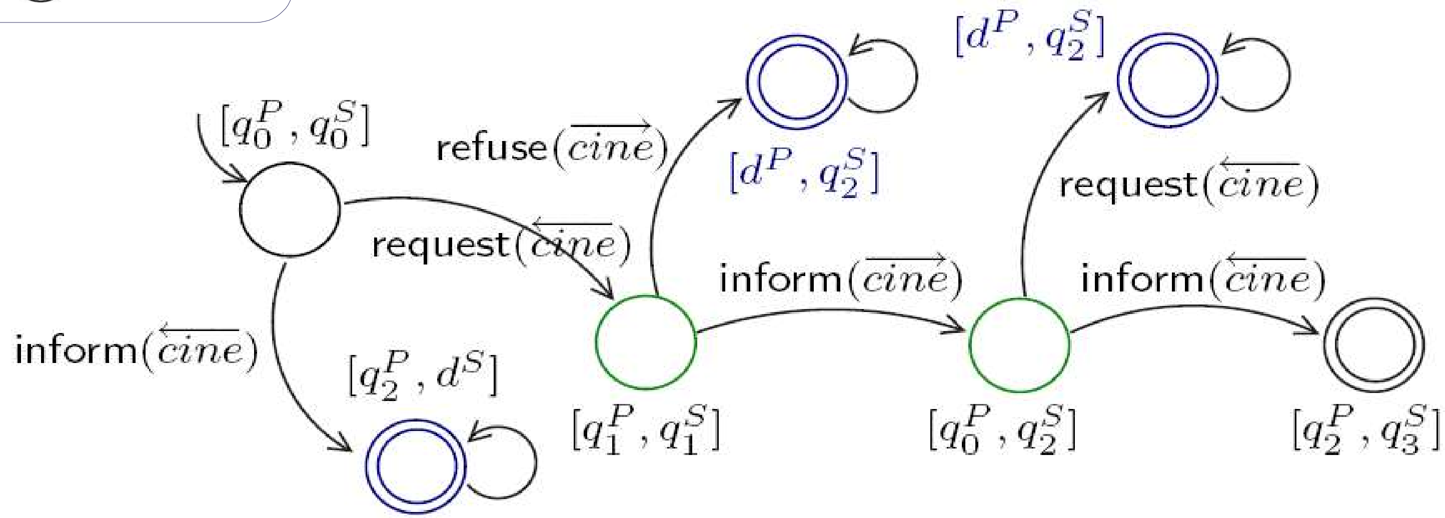
Translating BPEL4WS to FSM


inform($\overleftarrow{cine}$)
q_0^P
q_2^P
request($\overleftarrow{cine}$)
inform($\overrightarrow{cine}$)
q_1^P

The automaton M_{conf}


inform($\overleftarrow{cine}$)
q_0^P
q_2^P
request($\overleftarrow{cine}$)
inform($\overrightarrow{cine}$)
q_1^P


q_0^S
q_3^S
request($\overleftarrow{cine}$)
refuse($\overrightarrow{cine}$)
inform($\overleftarrow{cine}$)
q_1^S
inform($\overrightarrow{cine}$)
q_2^S


$[d^P, q_2^S]$
$[q_0^P, q_0^S]$
refuse($\overrightarrow{cine}$)
$[d^P, q_2^S]$
request($\overleftarrow{cine}$)
request($\overleftarrow{cine}$)
inform($\overrightarrow{cine}$)
inform($\overleftarrow{cine}$)
inform($\overleftarrow{cine}$)
$[q_2^P, d^S]$
$[q_1^P, q_1^S]$
$[q_0^P, q_2^S]$
$[q_2^P, q_3^S]$

- The automaton is complete and accepts both languages
- The agent's policy is conformant and interoperable

Final remarks

- An approach for verifying interoperability based on conformance test, that exploits the theory of formal languages
- There are other proposals for conformance tests in the literature but, to our knowledge, no demonstration that they guarantee interoperability is given
- Limits:
  - Two-party protocols
  - Regular languages
  - Infitine conversations