# Personalization, verification and conformance for logic-based communicating agents

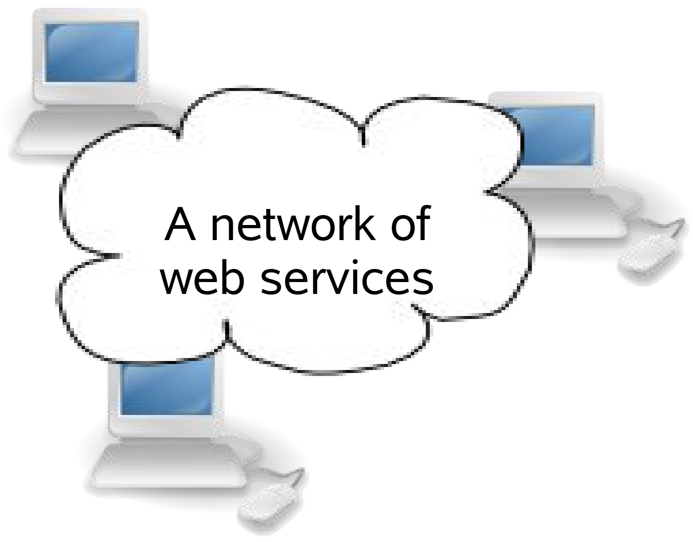
M. Baldoni, C. Baroglio, A. Martelli, V. Mascardi,
V. Patti, C. Schifanella, L. Torasso

## Main topics last two years in MASSiVE

- Extension of the language DyLOG by introducing a communication kit [AMAI 04, ICTCS 03]
- Personalization of courseware [AIRE 04; EAW 04; LNCS Tutorial 3564]
- Personalization of the interaction with web services [PPSWR 03; WS-FM 2004; JLAP 06, accepted after revision]
- Agent and Web services interoperability [CLIMA V; CLIMA VI, WS-FM 05]
- Integrated environments for agent-oriented software engineering [DALT 2004]

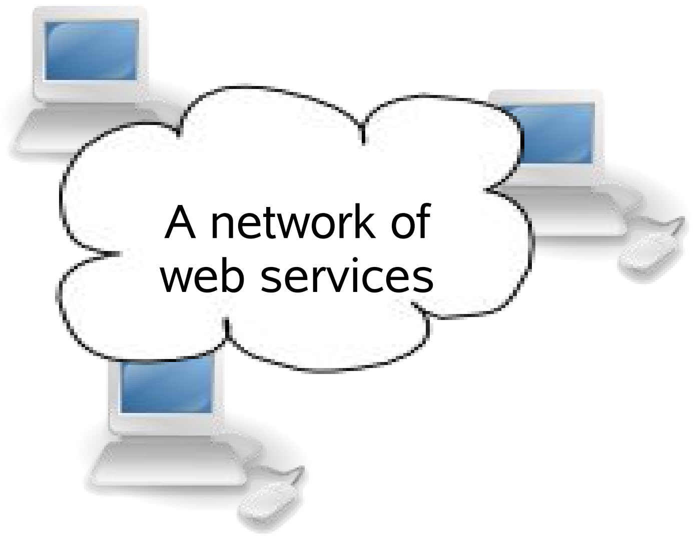
## Web services

- Web services are heterogeneous devices to be invoked over the web
- Executable description of their business process (especially the interactive behavior)
- Tasks: composition, selection, ...
- Dynamic dimension
- Web services share some similarities with agents



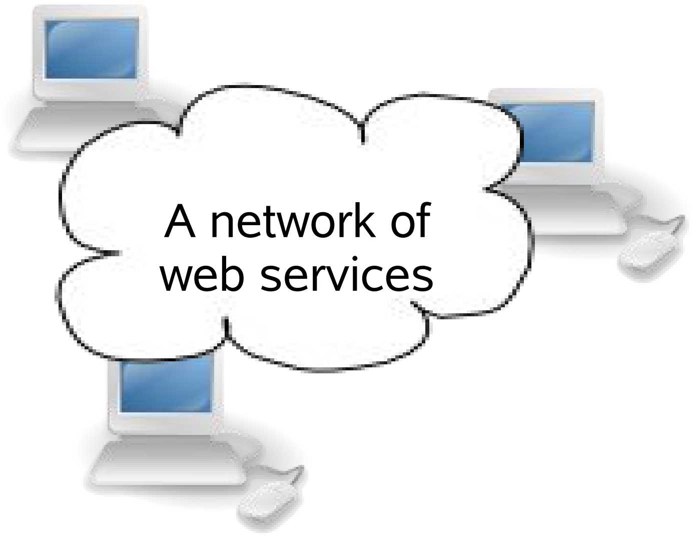
## Service-oriented multiagent systems

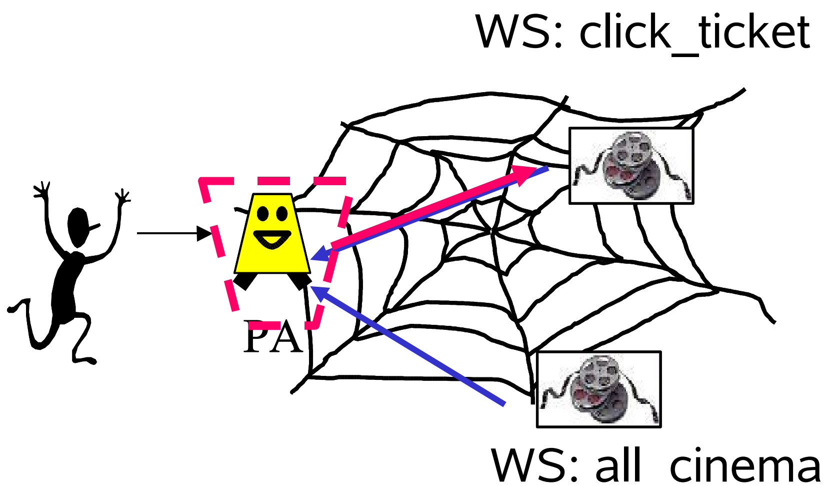
- Web services are heterogeneous devices to be invoked over the web
- Executable description of their business process (especially the interactive behavior)
- Tasks: composition, selection, ...
- Dynamic dimension
- Web services share some similarities with agents: ***service-oriented multiagent systems***



# User's needs and goals

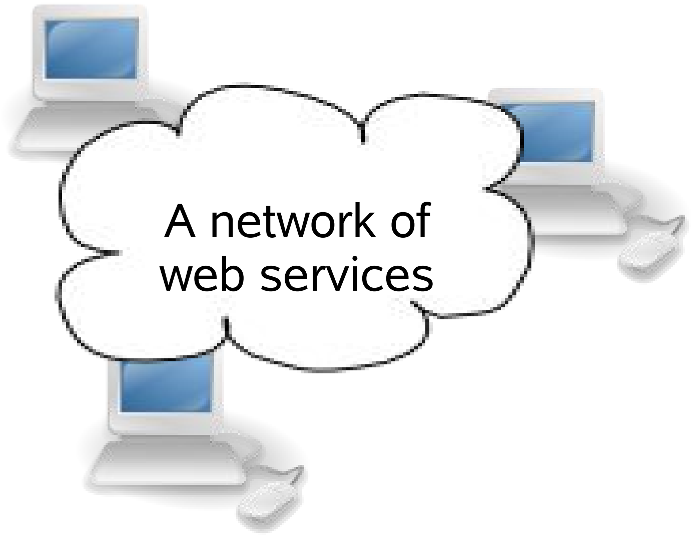
- Selection and composition: usually on the basis of general properties of the services themselves and of their interactive behavior (category, functional compositionality)
- User's needs and goals? Personalization of the access to the resources?
- They constraint the search
- Example: paying by credit card or by cash?

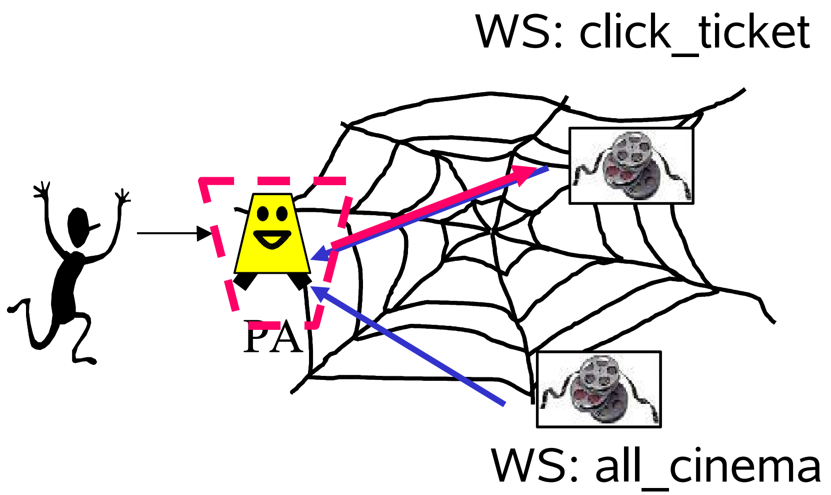




# Personalization of interaction

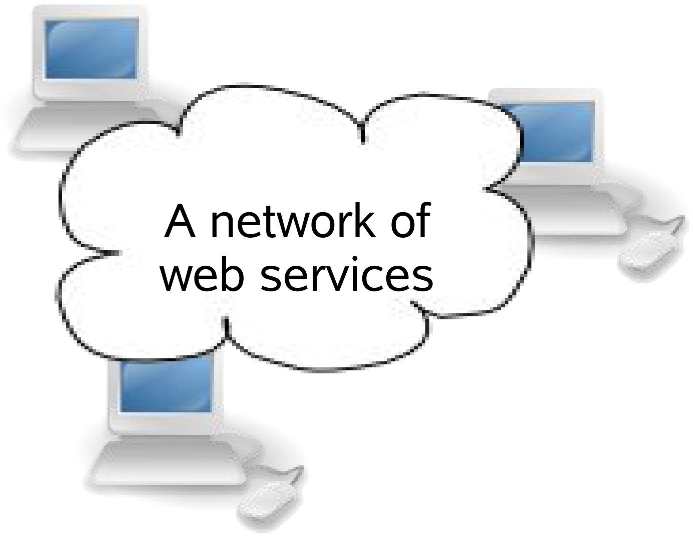
- Personalization is reasoning!
- Three necessary components:
  1. web services represented by means of some declarative formalism (with a well-defined semantics)
  2. automated tools for reasoning about such a description
  3. a representation of the user's requests
- These are missed in WS technologies [van der Aalst, WS-FM 05]

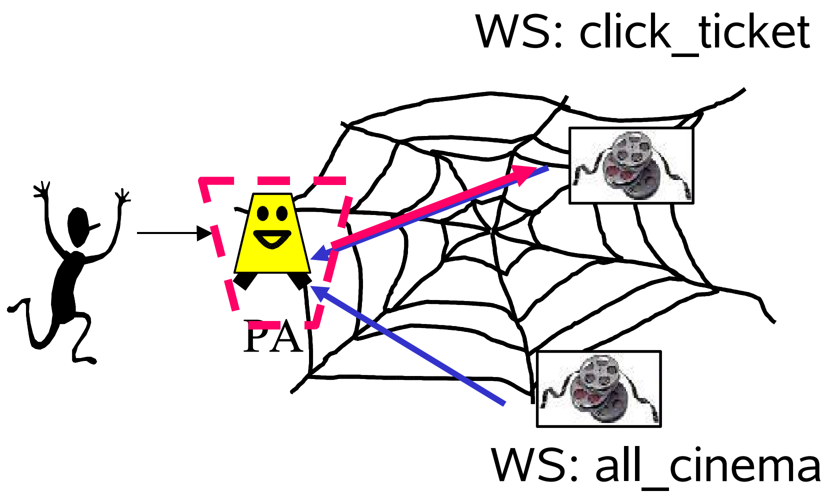




# Personalization of interaction [JLAP ??]

- Personalization by reasoning by actions and change: communicative actions and interaction protocols
- Reasoning about communicative behavior of the services: is it possible to make a deal with this service respecting the user's goals?
- In our proposal, logic programming reasoning techniques are used for understanding if the constraints of the customer fit in the policy of the service





## DyLOG + CKit [AMAI 04, ICTCS 03]

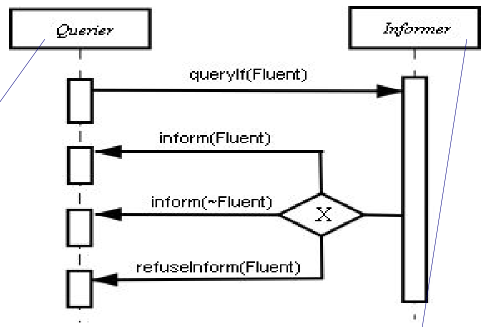
- A language to program agents, based on a *modal approach* for reasoning about actions and change
  - *Primitive actions*: preconditions and effects
  - *Sensing actions*: interaction with the world
  - *Prolog-like procedure definitions (complex actions)*: the agent's behavior

- A domain description is used to refer to a set of primitive action definitions, a set of sensing action definitions, a set of complex action definitions, together with a set of initial observations.

# Representing policies in DyLOG + CKit

- Agents have a *subjective perception* of communication with the others, then an agent represents a protocol as one of *its* (conversation) *policies*
- *Policies* are represented by a set of *inclusion axioms* of the form:

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



$$\langle \mathsf{yes\_no\_query}_Q(Self, Other, Fluent) \rangle \varphi \subset \langle \mathsf{queryIf}(Self, Other, Fluent); \mathsf{get\_answer}(Self, Other, Fluent) \rangle \varphi$$

$$[\mathsf{get\_answer}(Self, Other, Fluent)]\varphi \equiv [\mathsf{inform}(Other, Self, Fluent) \cup \mathsf{inform}(Other, Self, \neg Fluent) \cup \mathsf{refuseInform}(Other, Self, Fluent)]\varphi$$

$$\langle \mathsf{yes\_no\_query}_I(Self, Other, Fluent) \rangle \varphi \subset \langle \mathsf{get\_start}(Self, Other, Fluent); \mathcal{B}^{Self} Fluent?; \mathsf{inform}(Self, Other, Fluent) \rangle \varphi$$
$$\langle \mathsf{yes\_no\_query}_I(Self, Other, Fluent) \rangle \varphi \subset \langle \mathsf{get\_start}(Self, Other, Fluent); \mathcal{B}^{Self} \neg Fluent?; \mathsf{inform}(Self, Other, \neg Fluent) \rangle \varphi$$
$$\langle \mathsf{yes\_no\_query}_I(Self, Other, Fluent) \rangle \varphi \subset \langle \mathsf{get\_start}(Self, Other, Fluent); \mathcal{U}^{Self} Fluent?; \mathsf{refuseInform}(Self, Other, Fluent) \rangle \varphi$$
$$[\mathsf{get\_start}(Self, Other, Fluent)]\varphi \equiv [\mathsf{queryIf}(Other, Self, Fluent)]\varphi$$

## DyLOG + Ckit: overview

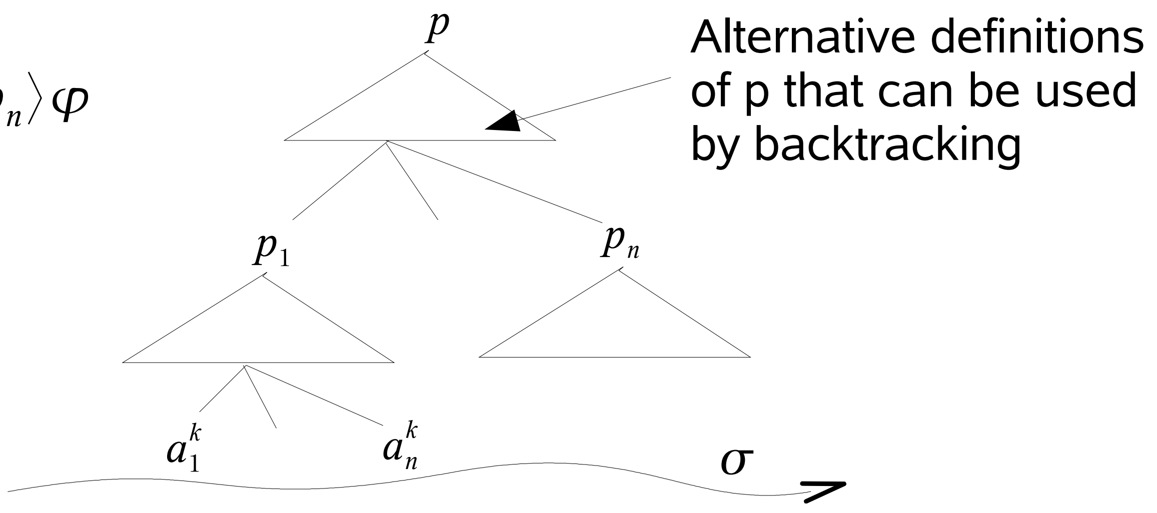
- Given a domain description, we can reason about it by means of *existential queries*:

$$(\Pi, CKit^{ag_i}, S_0) \vdash \langle p \rangle Fs \; w.a. \; \sigma$$

  - *p* is an interaction protocol
  - We look for a conversation, which is an *instance* of the policiy described by *p*, after which the condition *Fs* holds

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

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

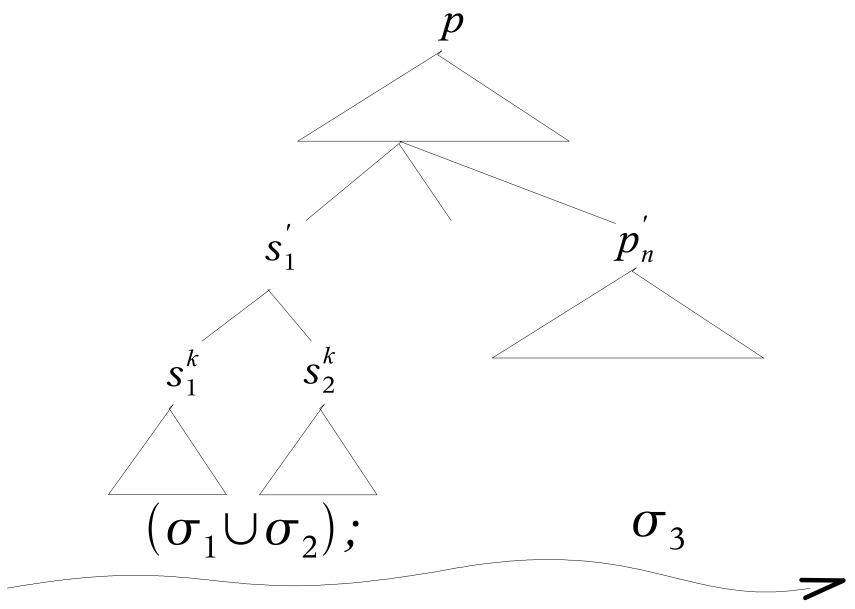

$p$
Alternative definitions of p that can be used by backtracking
$p_1$
$p_n$
$a_1^k$
$a_n^k$
$\sigma$

## DyLOG + Ckit: testing a priori

- Look for a protocol that has one possible execution, after which the service provider does not know the customer's credit card number, and a reservation has been taken

$$\langle search\_service(restaurant, Protocol)\ ;\ Protocol(customer, service, time)\rangle$$
$$(\mathcal{B}^{customer}\neg\mathcal{B}^{service} cc\_number \wedge \mathcal{B}^{customer} reservation(time))$$

Existential query!!

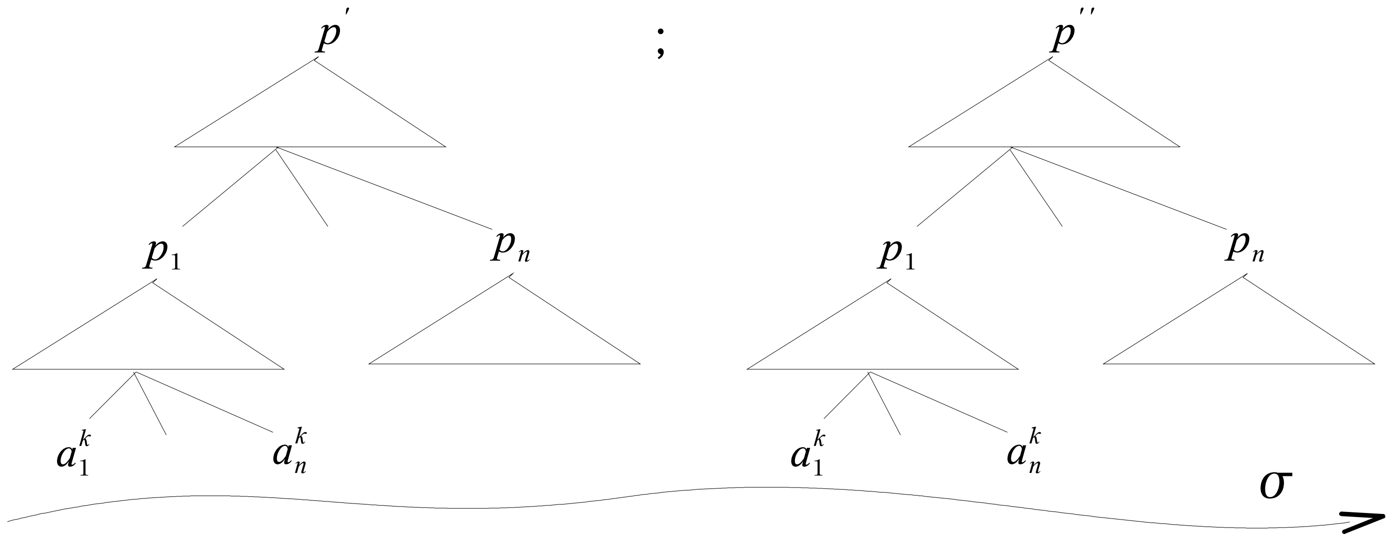


## DyLOG + Ckit: testing a priori

- Is it possible to compose the interaction so to reserve a table for dinner and to book a ticket for a movie, exploiting a promotion?

$$\langle \mathsf{reserv\_rest\_1}_C(customer, restaurant, dinner)\ ;\ \mathsf{reserv\_cinema\_1}_C(customer, cinema, movie)\rangle$$
$$(\mathcal{B}^{customer} cinema\_promo \wedge \mathcal{B}^{customer} reservation(dinner) \wedge \mathcal{B}^{customer} reservation(movie) \wedge \mathcal{B}^{customer}\mathcal{B}^{C} ft\_number)$$

Existential query!!



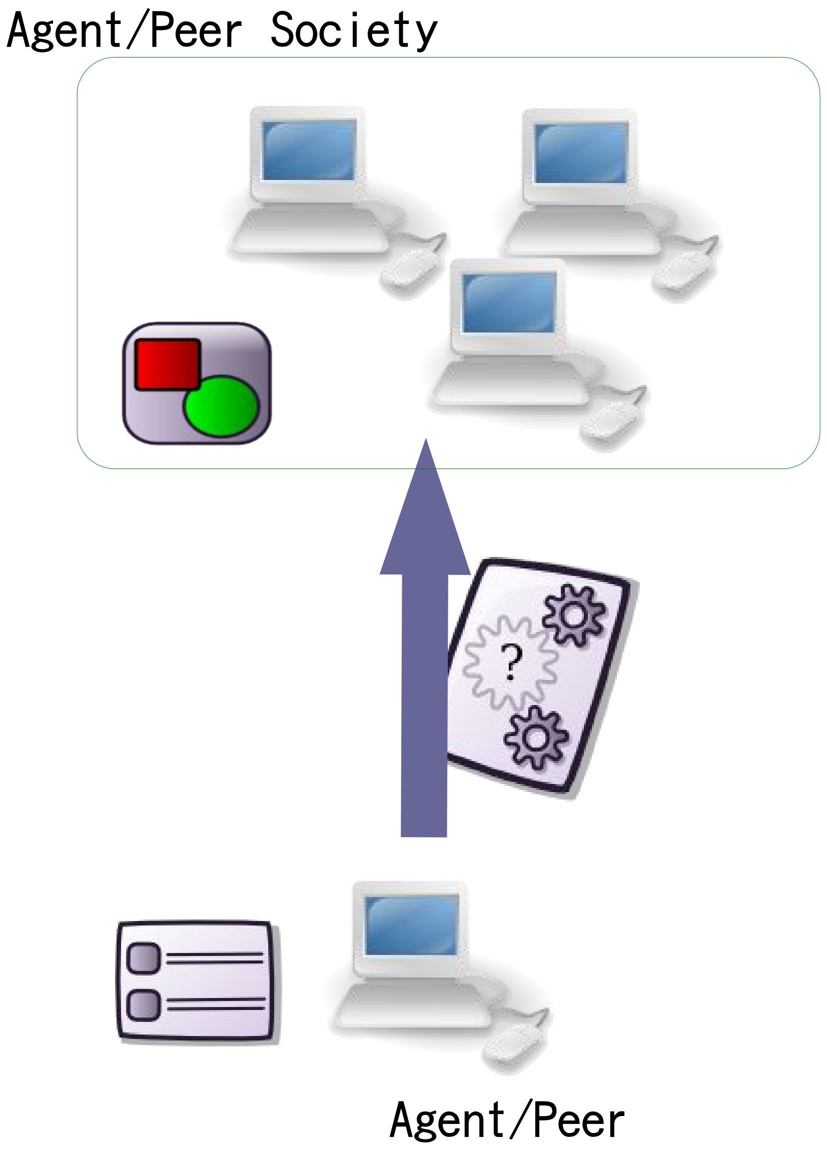
## Testing a priori: an important assumption

- Our proposal can be considered as a second step in the matchmaking, which narrows a set of already selected services and performs a customization of the interaction with them
- Our vision of the steps to be taken toward the realization:
  - *public description of the interaction protocols in the form of choreographies (e.g. WS-CDL-like descriptions)*
  - *download and translation in a declarative representation in order to perform the reasoning task*
- **Important assumption**: the implementation of the web service behavior (e.g. in a BPEL-like language) **must be conformant w.r.t. the protocol specification** that is used as input of the reasoning

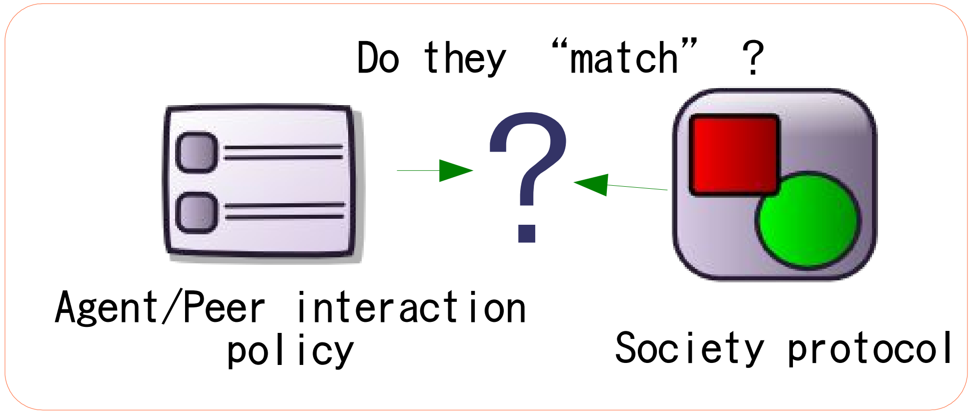
## The need for a global view of interaction

- Verification of the properties of the interactions among a set of processes: widely studied in in the literature
- The web adds a dynamic feature: the set of processes might evolve, with newcomers coming in at different timesteps
- There is a need for “distributing” this verification (also in time)
- Idea: add an abstract level!
- Define and make public the set of interaction rules that the group should follow (society protocol). A service can enter the society only if its interactive behavior conforms to the protocol
- The conformance test is to be conceived so as to preserve interoperability

# Checking interoperability



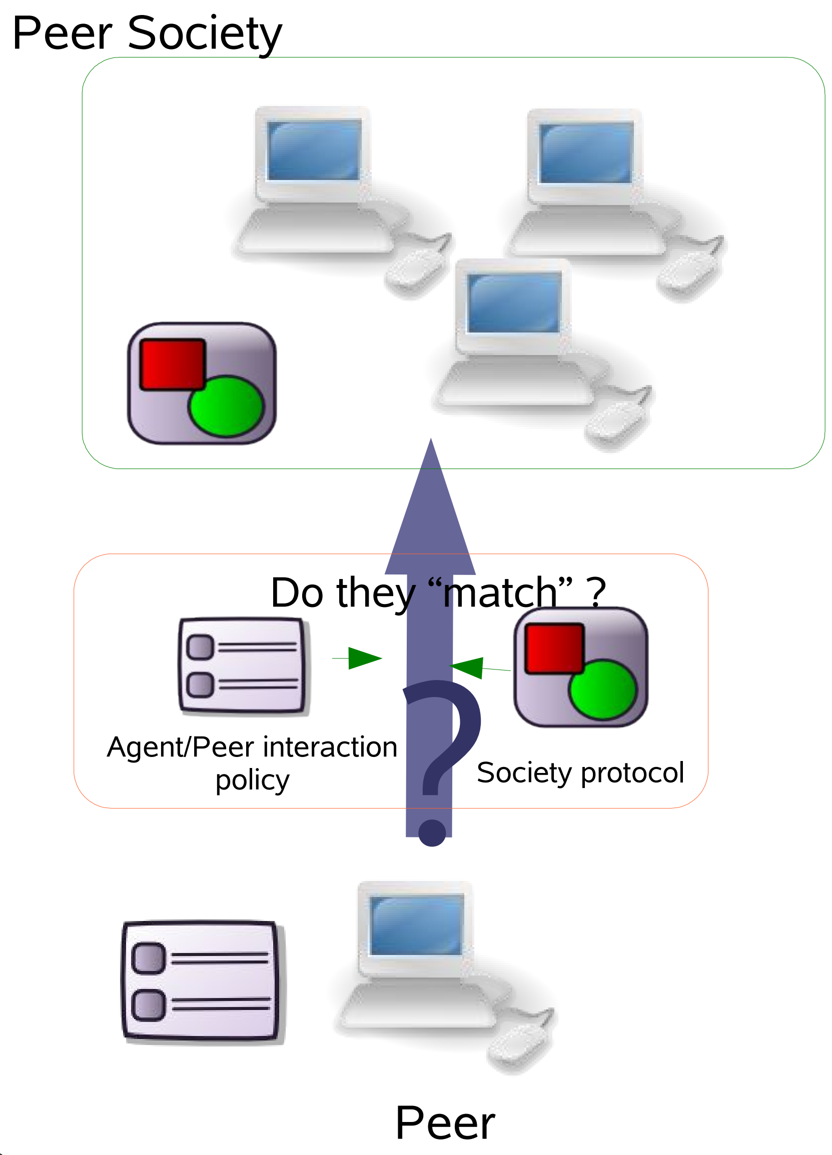
- Either we verify the interaction of each entity with each other
- Or we introduce a set of rules that determine the overall behavior: an interaction protocol
- Check the single peer's policy against society protocol



## The need for a global view of interaction

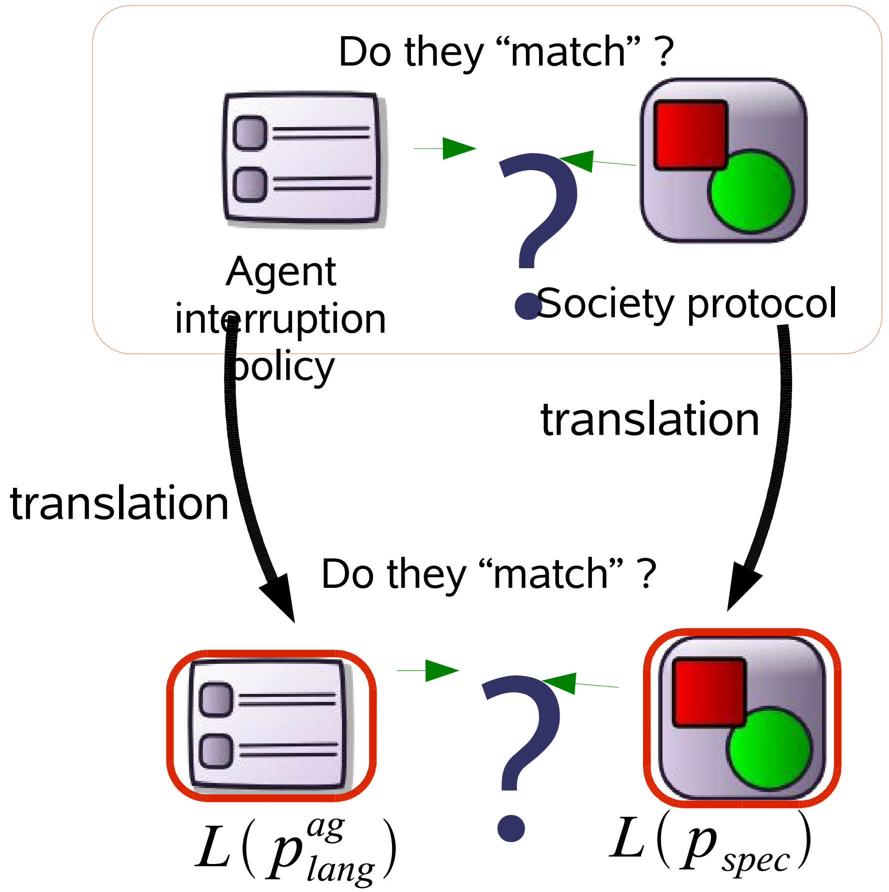
- The need for each service of an interface that is accessible through standard protocols and that ***describes the interaction capability*** of the service
- BPEL4WS
  - execution language: for specifying the actual behavior of a partecipant in a business interaction
  - modeling language: for specifying the interaction at an abstract level (from the perspective of the service being described)
- Capturing the behavior of BPEL in a formal way (process algebra, petri nets, FSM)
- Local point of view of the interaction is not sufficient!
- Choreography: WS-CDL (W3C proposal)

# 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-communicative behavior allowing execution, eg. BPEL executable process

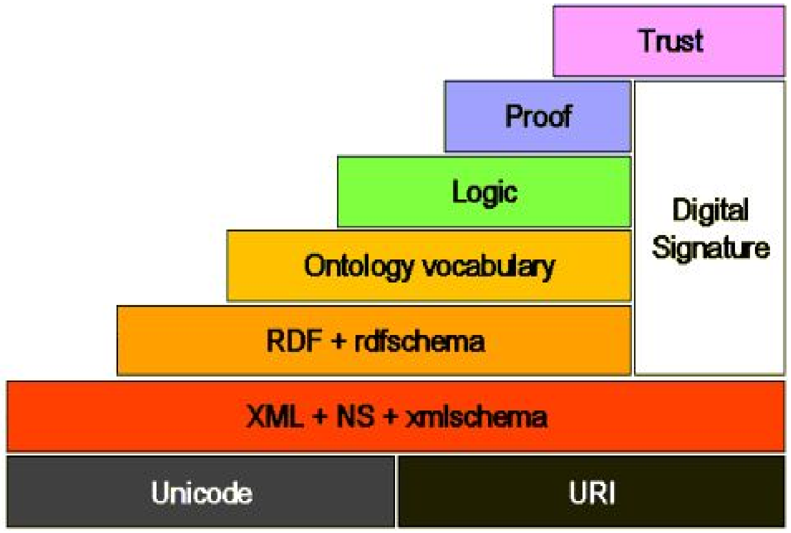
# A conformance test [CLIMA VI, WS-FM 05]



- 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

## Semantic Web (W3C) and OWL-S

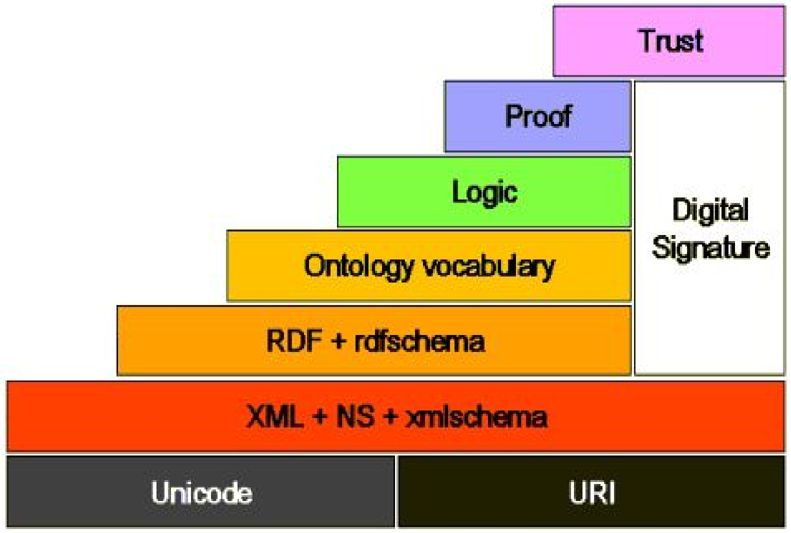
- Providing a common framework, that allows resources to be shared and reused across application, enterprise, and community boundaries:
  - Machine-processable
  - Declarative format
- Web services: OWL-S, description of possibly composite processes from a local perspective



Semantic Web Tower

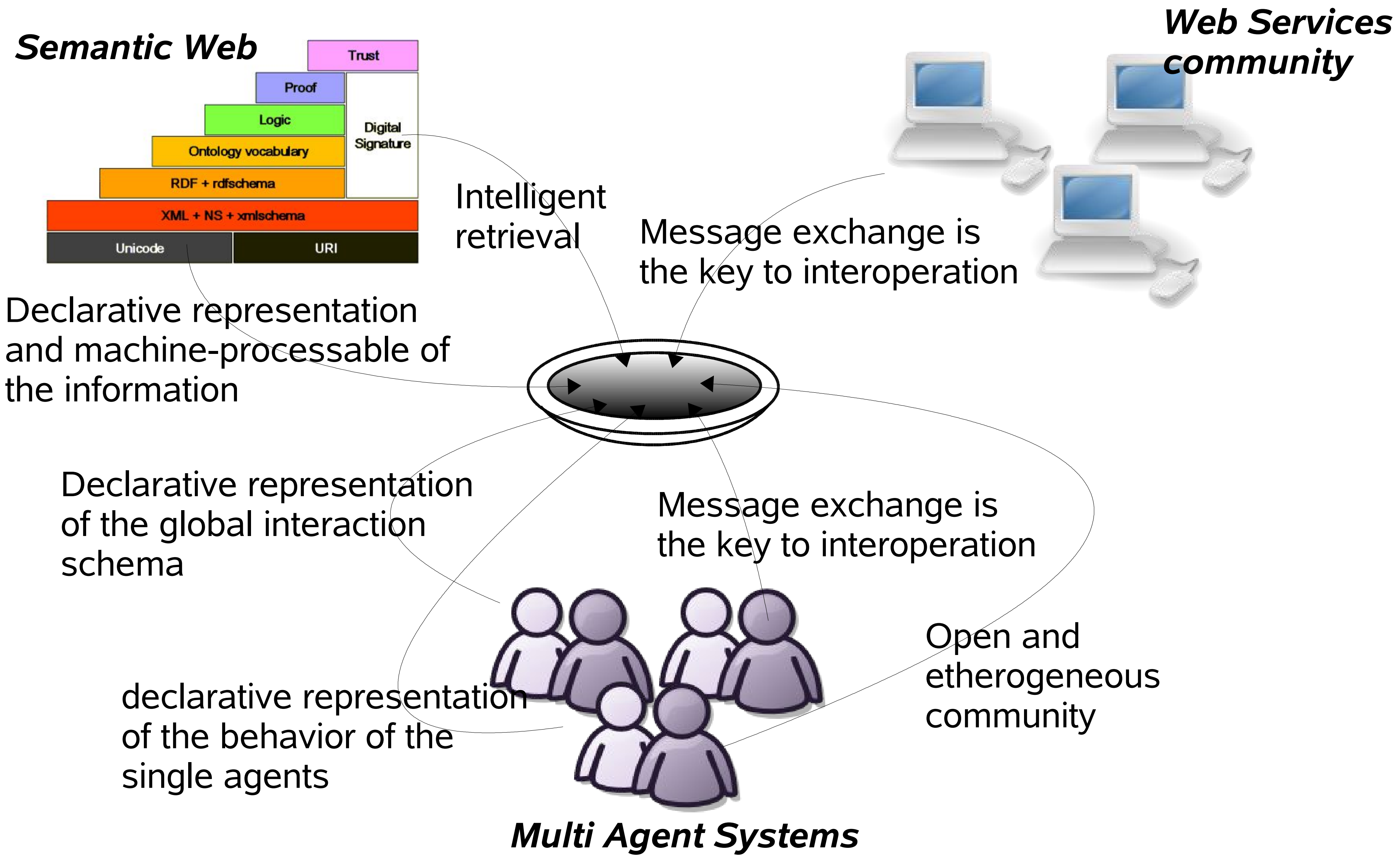
## Semantic Web (W3C) and OWL-S

- OWL-S: focussed on the **process advertisement** and the **process structure**
- There is currently not proposal of a concept close to that of “choreography”
- The proposed matchmaking techniques are still simple and quite far from fully exploiting the power of sharable semantics



Semantic Web Tower

# Semantic Web <-> Web Services <-> MAS



***Semantic Web***

***Web Services community***

***Multi Agent Systems***

Intelligent retrieval

Message exchange is the key to interoperation

Declarative representation and machine-processable of the information

Declarative representation of the global interaction schema

Message exchange is the key to interoperation

Open and etherogeneous community

declarative representation of the behavior of the single agents

## Back to Web Services

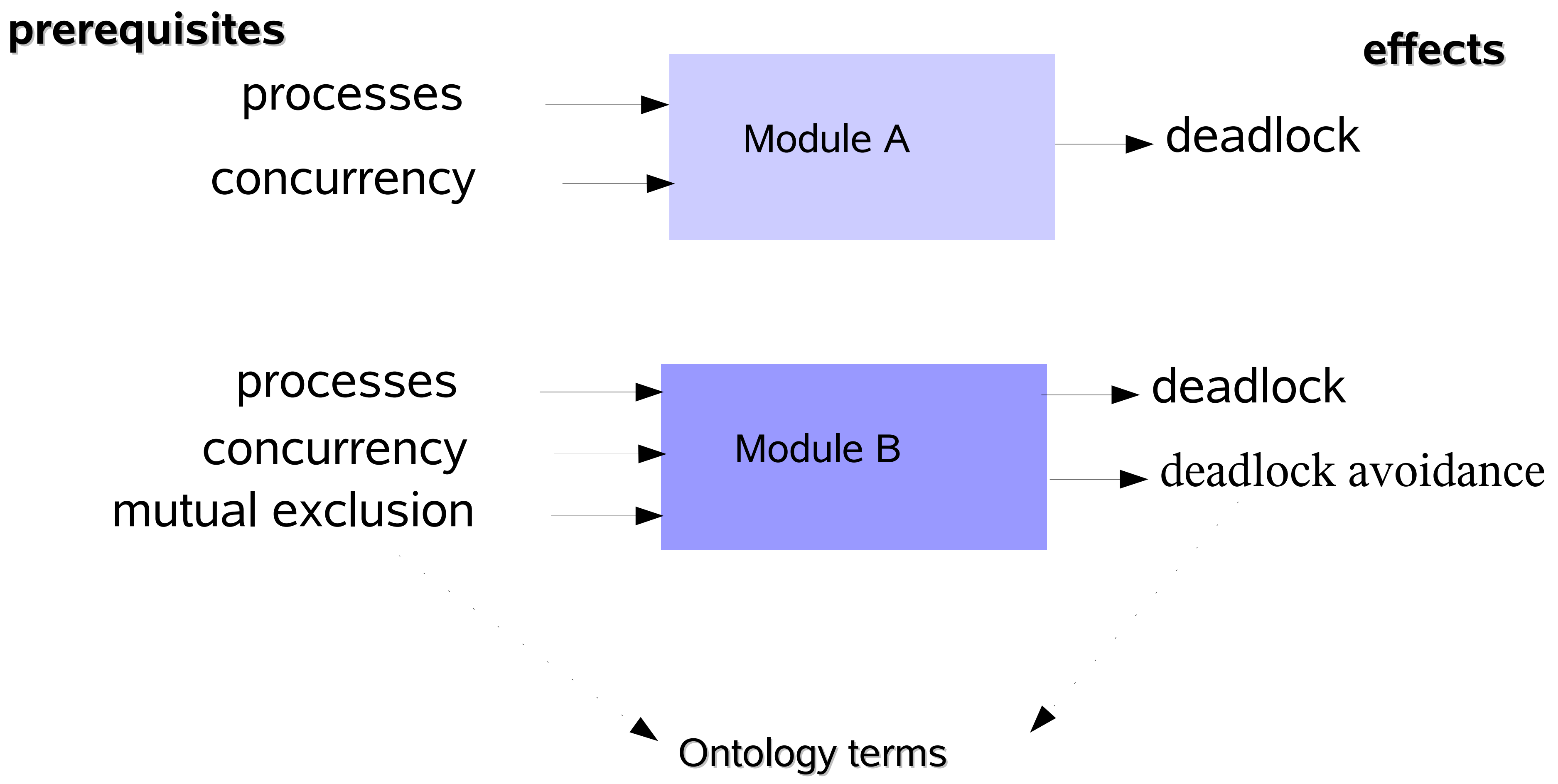
- A white box approach in which part of the behavior of the services is available for a rational inspection
- We do not deal with failure (at the execution time). Replanning, compensation techniques should be considered
- A declarative representation of the choreography and orchestration independent from the implementation language (BPEL is now a standard but tomorrow?)
- Very hot topics!

# DALT 2006 (in conjunction with AAMAS 2006)

- ***4th Int. Workshop on Declarative Agent Languages and Technologies, Hakodate, Japan 8 or 9 May 2006***
- **Important dates**:
  - **Submission deadline:** ***15 January 2006***
  - Notification of authors: 19 *February 2006*
  - Final version due: *8 March 2006*
  - **Workshop: *8 or 9 May 2006***

- "[…] This year we have *explicitly included the semantic web* amongst the topics of interest […] There is a need for formal tools for representing knowledge and mechanisms to reason about it. In particular, in the case of *semantic web services* there is also a need of representing autonomous entities that should automatically be retrieved, invoked, and composed so as to accomplish goals of interest. *This field is therefore a big opportunity for the research community working on declarative languages and on agents* […]"
- Among the topics:
  - ***agents and the semantic web***
  - ***service-oriented multiagent systems***

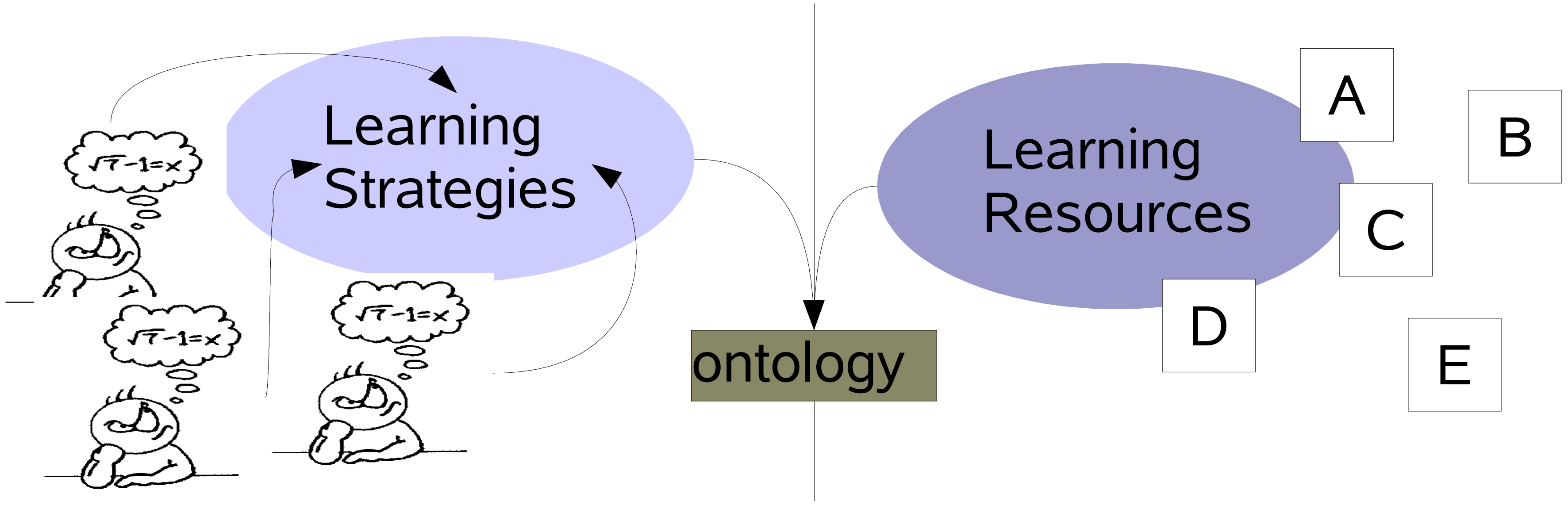
## Learning resources as actions [AIRE 04]



***Resources explicitly annotated by ontology terms!***

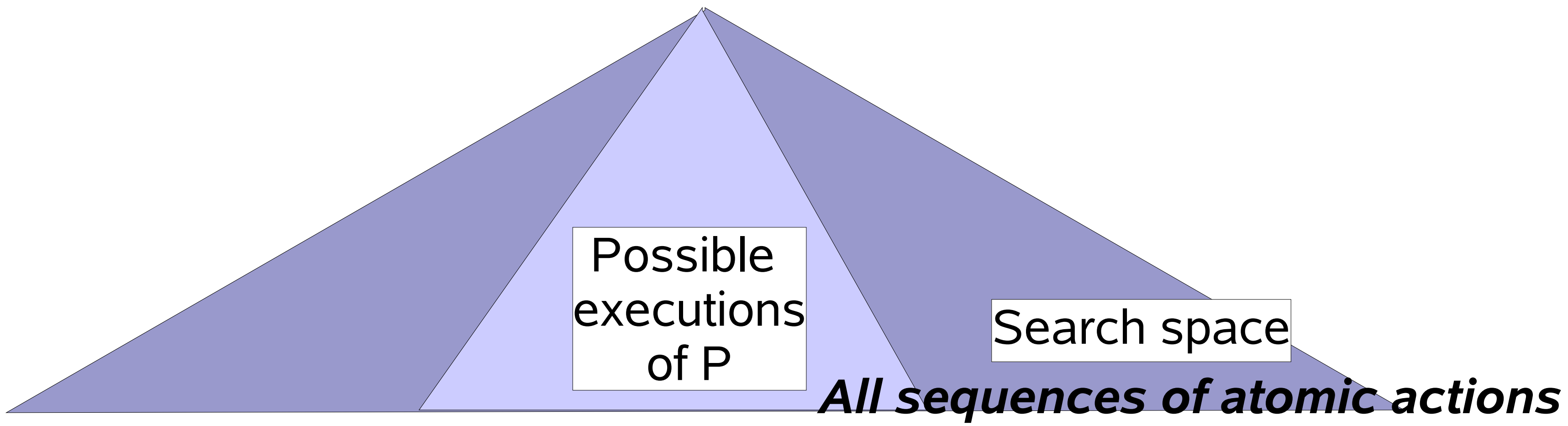
# Learning strategies

- The organization of the material in a lesson or a course is not only up to prerequisites and effects but also to the experience of the lecturer
- Learning strategy: overall schedule of the topics the view of teacher of how topics should be sequenced



# Procedural planning

- The search space is constrained by allowing only sequences of actions that are executions of a given procedure
  - plan: procedure execution
  - procedure:  behavior schema

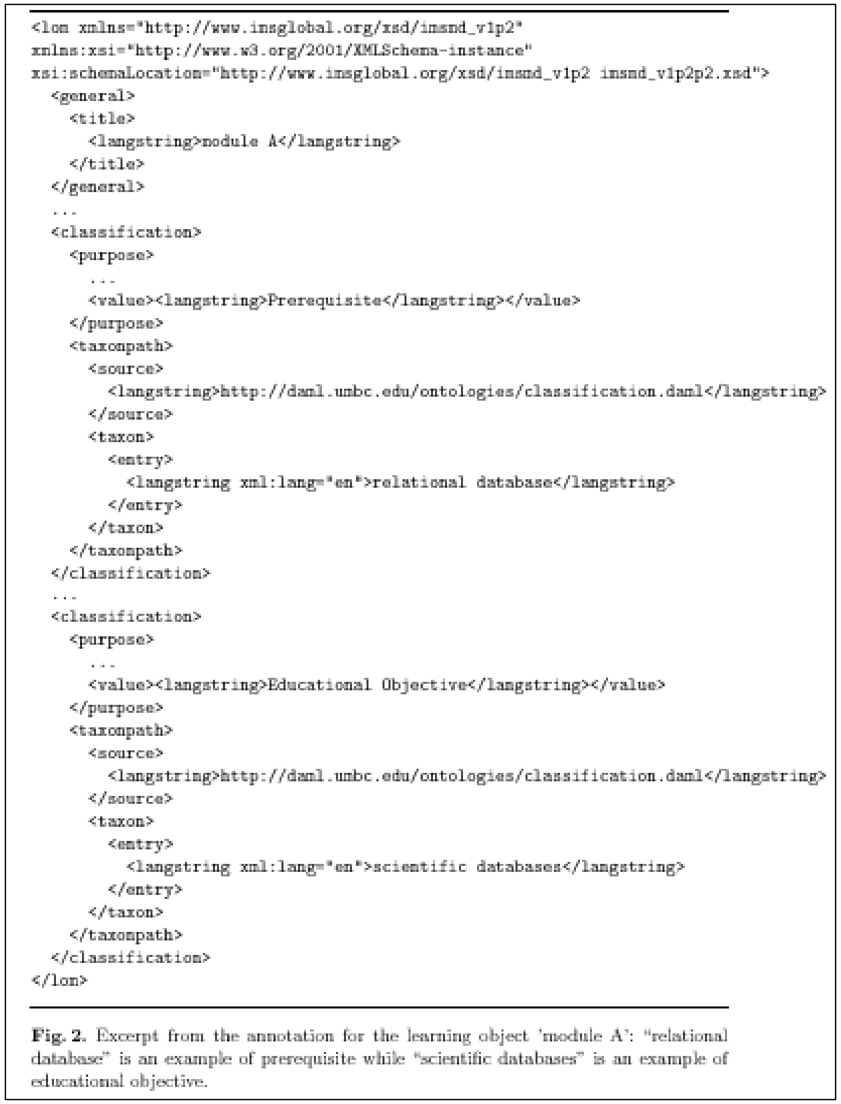


## SCORM

- Frameworks using standard learning object metadata: there already exist various proposals for standardizing the description of learning objects, to make them cross-platform (cross-LMS, learning management systems)
- One of the most interesting frameworks is **SCORM** (http://www.adlnet.org/)
- Why SCORM?
  - it is a standardized framework,
  - it describes LO's (IEEE LOM - Learning Tech. Standard committee), and
  - it also rules their presentation into a course

## Annotating LOM at the knowledge level [EAW 04]

- LOM: a complete LOM description consists of attributes;
- Attributes: nine categories (general, life cycle, meta-metadata, technical, educational, rights, relation, **classification**, and annotation)
- Annotating LOM at the knowledge level
- Classification attribute: includes the possibility of describing the contents of a learning object in terms of keywords taken from an ontology of interest -> by means of LOM it is possible to include in a SCO a description at the level of knowledge entities

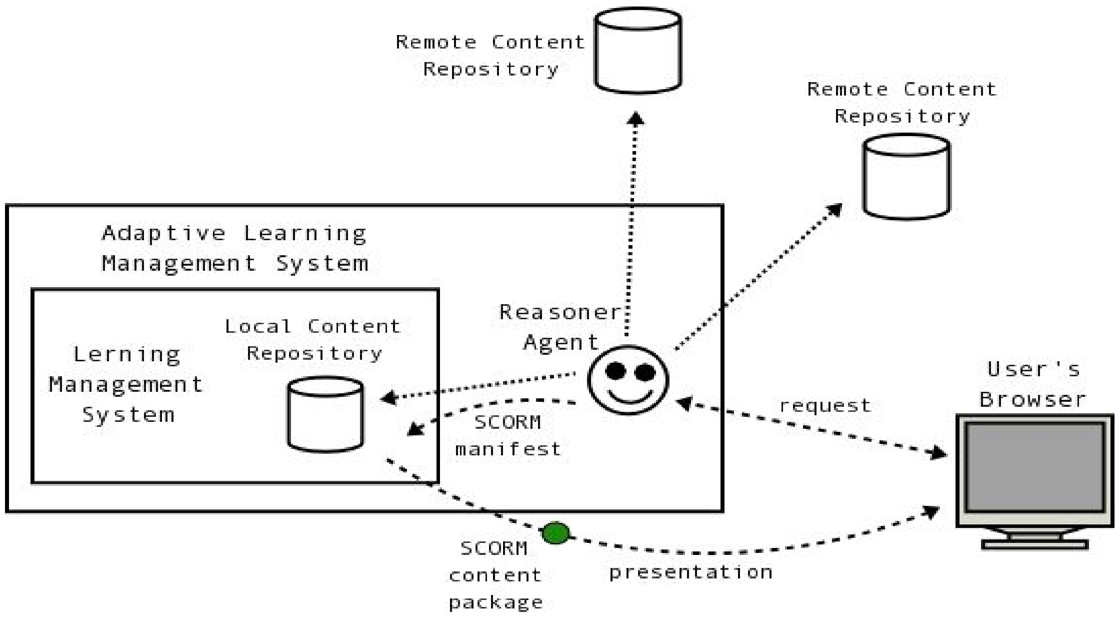


```
<lom xmlns="http://www.imsglobal.org/xsd/imsmd_v1p2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imsmd_v1p2 imsmd_v1p2p2.xsd">
  <general>
    <title>
      <langstring>module A</langstring>
    </title>
  </general>
  ...
  <classification>
    <purpose>
      ...
      <value><langstring>Prerequisite</langstring></value>
    </purpose>
    <taxonpath>
      <source>
        <langstring>http://daml.umbc.edu/ontologies/classification.daml</langstring>
      </source>
      <taxon>
        <entry>
          <langstring xml:lang="en">relational database</langstring>
        </entry>
      </taxon>
    </taxonpath>
  </classification>
  ...
  <classification>
    <purpose>
      ...
      <value><langstring>Educational Objective</langstring></value>
    </purpose>
    <taxonpath>
      <source>
        <langstring>http://daml.umbc.edu/ontologies/classification.daml</langstring>
      </source>
      <taxon>
        <entry>
          <langstring xml:lang="en">scientific databases</langstring>
        </entry>
      </taxon>
    </taxonpath>
  </classification>
</lom>
```

**Fig. 2.** Excerpt from the annotation for the learning object 'module A': "relational database" is an example of prerequisite while "scientific databases" is an example of educational objective.

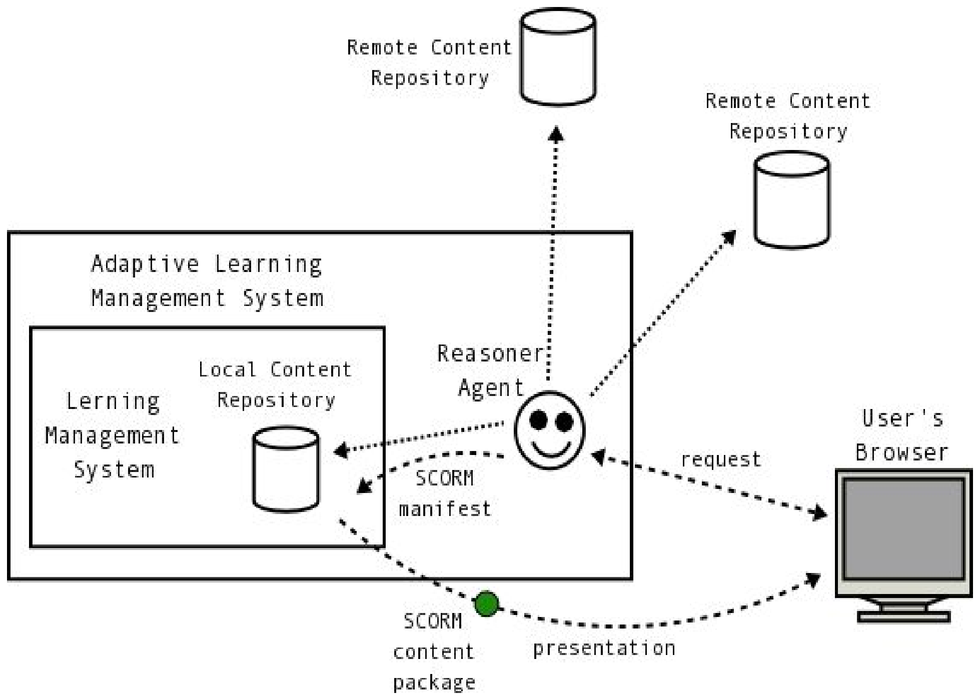
## Extending LMS

- All these steps should be carried on by the intelligent component added to the LMS architecture
- The resulting plan can be stored as a SCORM manifest, which can be considered as an instance of the original learning strategy


Remote Content Repository
Remote Content Repository
Adaptive Learning Management System
Lerning Management System
Local Content Repository
Reasoner Agent
SCORM manifest
request
User's Browser
SCORM content package
presentation

# Learning strategies and verification

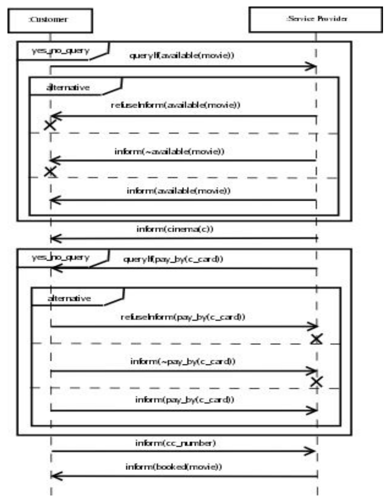
- Given a SCORM representation of a course:
- Is a SCORM Manifest conform to the learning strategies given by the teacher?
- Is the current course presentation conform to the learning strategies given by the teacher?



Remote Content Repository
Remote Content Repository
Adaptive Learning Management System
Lerning Management System
Local Content Repository
Reasoner Agent
User's Browser
request
SCORM manifest
SCORM content package
presentation

# Translation: an example [CLIMA V]

extract



AUML interaction diagram

***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

DyLOG implementation

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

***Sequences corresponding to all possible dialogues allowed by the implementation***

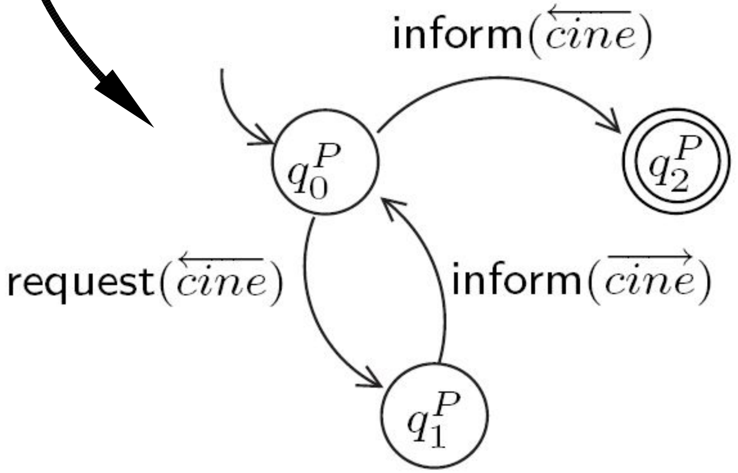
extract

## 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($\overrightarrow{cine}$)
$q_1^P$

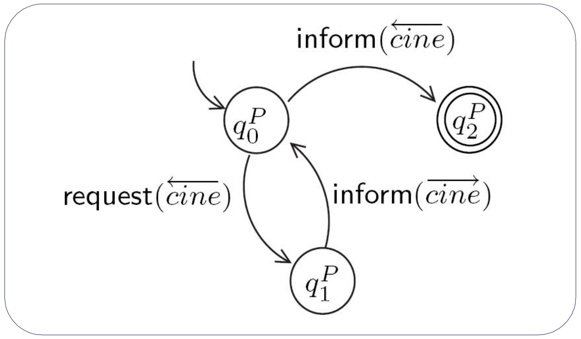
- This can be done by algorithm 2 of CLIMA V
- It exploits the form of inclusion axioms used to encode conversation policies:

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

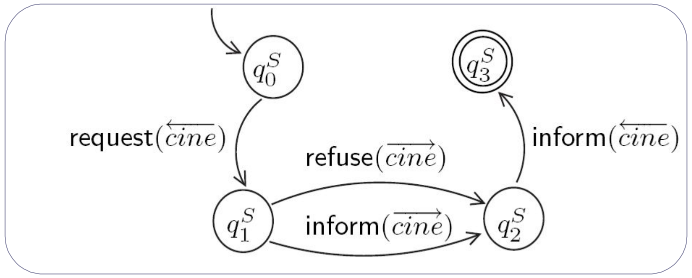
$$\Downarrow$$

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

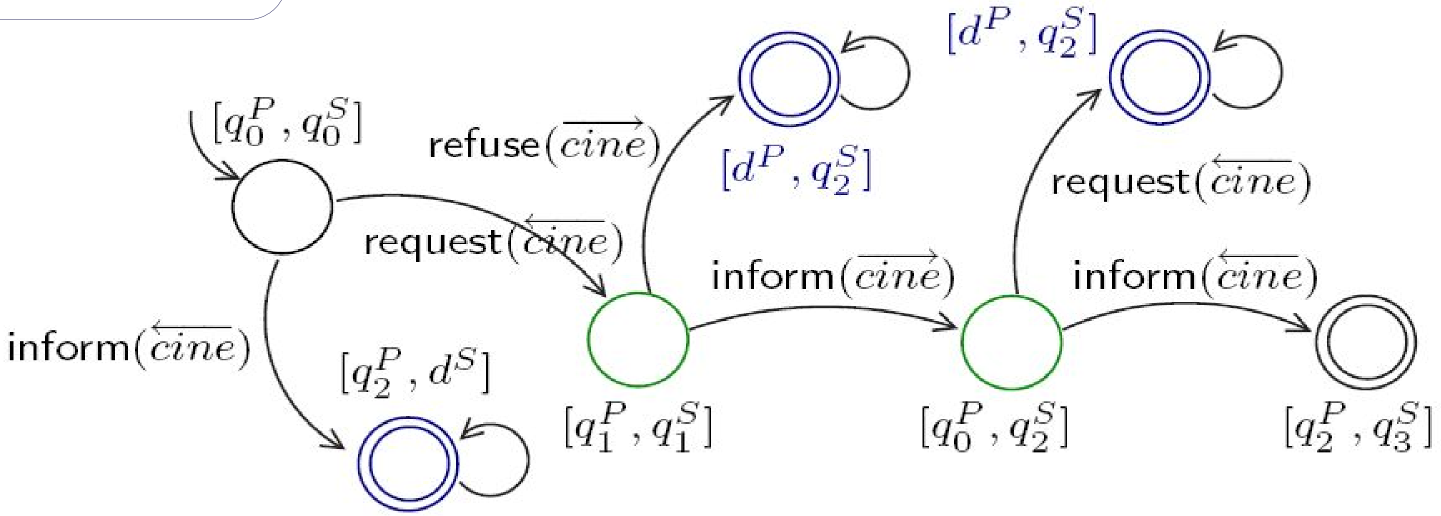
## The automaton $M_{conf}$



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($\overrightarrow{cine}$)
$q_2^S$



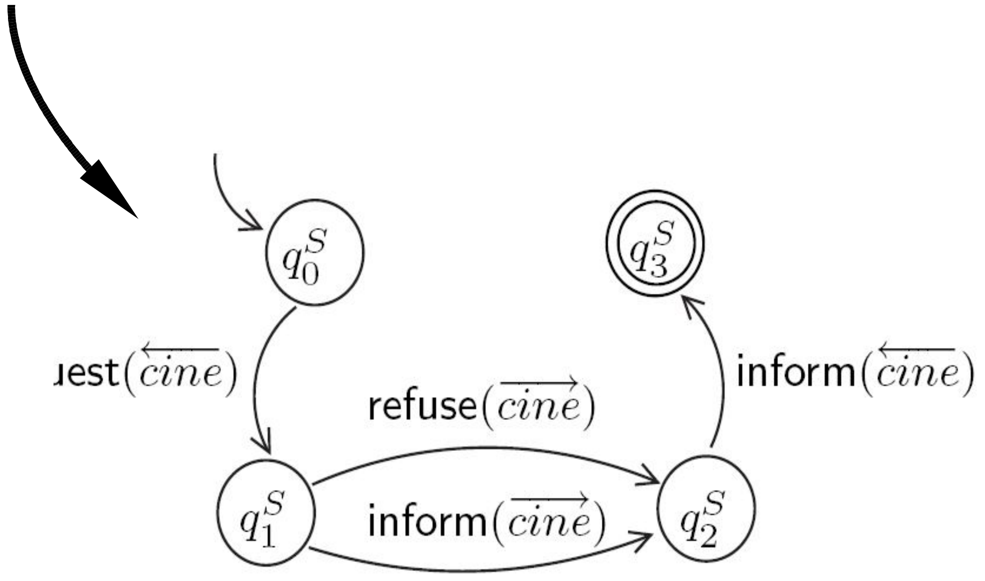
$[q_0^P, q_0^S]$
refuse($\overrightarrow{cine}$)
request($\overleftarrow{cine}$)
$[d^P, q_2^S]$
$[d^P, q_2^S]$
request($\overleftarrow{cine}$)
inform($\overrightarrow{cine}$)
inform($\overleftarrow{cine}$)
inform($\overleftarrow{cine}$)
$[q_2^P, d^S]$
$[q_1^P, q_1^S]$
$[q_0^P, q_2^S]$
$[q_2^P, q_3^S]$

- 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>
```

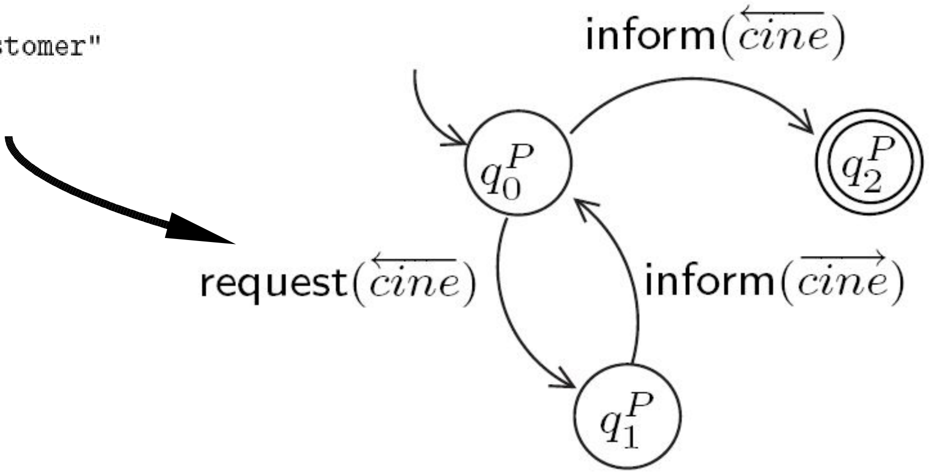
Translating WS-CDL to FSM


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

## Policy translation: BPEL4WS to FSM

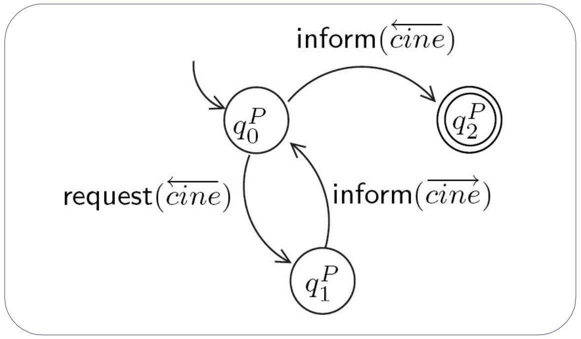
```
<sequence>
   ...
   <while condition="bpws:getVariableData('done') = 'false'">
      <pick>
         <onMessage portType="movieInfoPT" partnerLink="customer"
            operation="movieInfoACK">
            <assign>
               <copy>
                  <from expression="true" />
                  <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

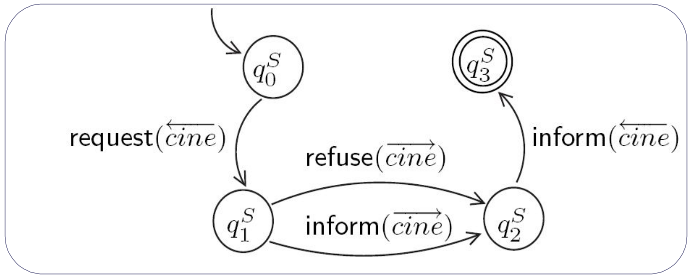


$\mathsf{inform}(\overleftarrow{cine})$
$q_0^P$
$q_2^P$
$\mathsf{request}(\overleftarrow{cine})$
$\mathsf{inform}(\overrightarrow{cine})$
$q_1^P$

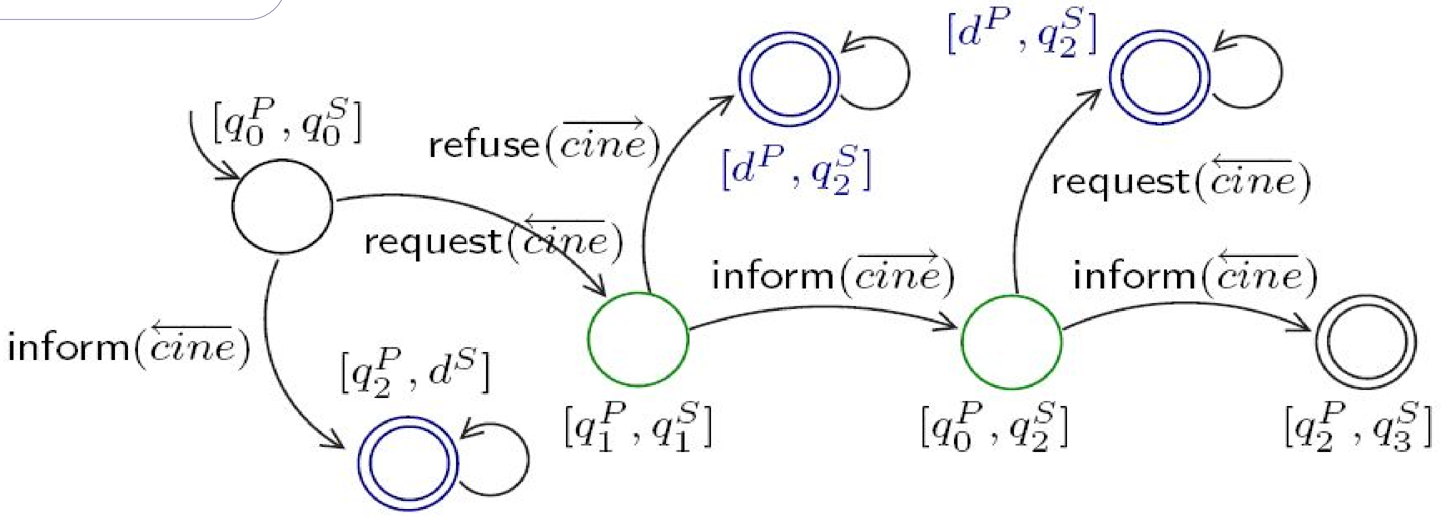
# The automaton $M_{conf}$



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($\overrightarrow{cine}$)
$q_2^S$



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